Guide du développeur

AWS IoT Core

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

AWS IoT Core: Guide du développeur

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

Les marques commerciales et la présentation commerciale d'Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d'une manière susceptible d'entraîner une confusion chez les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce que c'est AWS IoT ? 1
Comment vos appareils et applications accèdent AWS IoT 2
Ce que AWS IoT je peux faire 3
L'IoT dans l'industrie 3
L'IoT dans la domotique 3
Comment AWS IoT fonctionne 4
L'univers de l'IoT Side 4
AWS IoT aperçu des services 7
AWS IoT Core services 12
En savoir plus sur AWS IoT 17
Ressources de formation pour AWS IoT 17
AWS IoT ressources et guides 17
AWS IoT sur les réseaux sociaux 18
AWS services utilisés par le moteur de AWS IoT Core règles 19
Protocoles de communication pris en charge par AWS IoT Core 20
Nouveeeeeeautés dans la nouvelleAWS IoT clé de 21
Légende 24
Travailler avec AWS SDKs 24
Didacticiel de démarrage 26
Connectez votre premier appareil à AWS IoT Core 26
Configurez Compte AWS 28
Inscrivez-vous pour un Compte AWS 28
Création d’un utilisateur doté d’un accès administratif 29
Ouvrez la AWS IoT console 30
Tutoriel interactif 31
Connexion d'appareils IoT 32
Enregistrer l'état de l'appareil hors ligne 33
Acheminement des données de l'appareil vers les services 34
Tutoriel de connexion rapide 35
Étape 1. Commencez avec le didacticiel 36
Étape 2. Cela crée un objet 37
Étape 3. Téléchargez des fichiers sur votre appareil 41
Étape 4 : Exécutez l'échantillon 44
Étape 5. Explorez davantage 47

Testez la connectivité 48
Tutoriel de connexion avancé 54
Quelle option d'appareil vous convient le mieux ? 55
Créez des AWS IoT ressources 56
Configurer votre appareil 61
Afficher les messages MQTT avec le client AWS IoT MQTT 102
Affichage des messages MQTT dans le client MQTT 103
Publication de messages MQTT à partir du client MQTT 105
Test des abonnements partagés dans le client MQTT 107
AWS IoT tutoriels 110
Création de démos avec le AWS IoT Device Client 110
Conditions préalables à la création de démos avec le Device Client AWS IoT 111
Préparation à l'utilisation du client IoT Device 114
Installation et configuration d'un client pour appareils IoT 129
Communiquez avec le client Device à l'aide de MQTT 142
Exécutez des tâches IoT avec le Device Client 163
Nettoyage 178
Création de solutions à l'aide de l' AWS IoT appareil SDKs 188
Commencez à créer des solutions avec l' AWS IoT appareil SDKs 188
Connexion d'un appareil à AWS IoT Core l'aide de l' AWS IoT appareil SDK 189
Création de AWS IoT règles pour acheminer les données des appareils vers d'autres services 214
Conservation de l'état de l'appareil lorsque celui-ci est hors ligne avec Device Shadows 261
Création d'un autorisateur personnalisé pour AWS IoT Core 292
Surveillance de l'humidité du sol avec un AWS IoT Raspberry Pi 311
Connect à AWS IoT Core 326
AWS IoT Core- extrémités du point de terminaison du plan de contrôle 327
AWS IoT points de terminaison de l'appareil 328
AWS IoT Core pour les passerelles et appareils LoRa WAN 329
Connectez-vous aux points de terminaison AWS IoT Core du service 330
AWS CLI pour AWS IoT Core 331
AWS SDKs 332
AWS Applications mobiles SDKs 337
RESTE APIs des AWS IoT Core services 338
Connectez des appareils à AWS IoT 339
AWS IoT données de l'appareil et points de terminaison de service 340

AWS IoT Device SDKs ... 342
Protocoles de communication des appareils ... 345
Rubriques MQTT ... 388
Configurations de domaine ... 417
Connectez-vous aux points de AWS IoT terminaison FIPS ... 447
AWS IoT Core- extrémités du point de terminaison du plan de contrôle ... 447
AWS IoT Core - Points de terminaison du plan de données ... 448
AWS IoT Core- points de terminaison du fournisseur d'informations d'identification ... 449
Points de terminaison de données sur les tâches AWS IoT Device Management - ... 449
AWS IoT Device Management - Points de terminaison Fleet Hub ... 450
AWS IoT Device Management - points de terminaison sécurisés pour le tunneling ... 450
AWS IoT Device Management - Points de terminaison d'intégrations gérés ... 450
Gérer les appareils ... 452
Registre ... 453
Créer un objet ... 453
Liste des objets ... 454
Décrivez des objets ... 456
Mettre à jour un objet ... 457
Supprimer un objet ... 457
Attacher un mandataire à un objet ... 458
Énumérer les éléments associés à un directeur ... 459
Répertorier les principes associés à un objet ... 459
Lister les éléments associés à une V2 principale ... 460
Répertorier les principes associés à un objet V2 ... 461
Détacher un mandataire d'un objet ... 461
Types d'objets ... 462
Créer un type d'objet ... 463
Liste des types d'objets ... 463
Décrire un type d'objet ... 464
Associer un type d'objet à un objet ... 464
Mettre à jour un type d'objet ... 465
Rendre obsolète un type d'objet ... 465
Supprimer un type d'objet ... 467
Groupes d’objets statiques ... 467
Créer un groupe d’objets statiques ... 469
Décrire un groupe d'objets ... 470

Ajouter un objet à un groupe d’objets statiques 471
Supprimer un objet d'un groupe d’objet statiques 472
Répertorier les objets d'un groupe d'objets 472
Répertorier les groupes d'objets 473
Répertorier les groupes d'un objet 475
Mettre à jour un groupe d’objets statiques 476
Supprimer un groupe d'objets 476
Attacher une stratégie à un groupe d’objets statiques 477
Détacher une stratégie d'un groupe d’objets statiques 478
Répertorier les stratégies attachées à un groupe d'objets statiques 478
Répertorier les groupes d'une stratégie 478
Obtenir des stratégies efficaces pour un objet 479
Tester l'autorisation pour les actions MQTT 480
Groupes d'objets dynamiques 482
Cas d'utilisation de groupes d'objets dynamiques 482
Créer un groupe d'objets dynamique 484
Décrire un groupe d'objets dynamique 485
Mettre à jour un groupe d'objets dynamique 486
Supprimer un groupe d'objets dynamique 487
Limitations des groupes d'objets dynamiques et statiques 487
Limites relatives aux groupes d'objets dynamiques 488
Associer un objet à une connexion 490
Cas d’utilisation 491
Comment associer un objet à une connexion 492
Ajouter des attributs de propagation 495
AWS Management Console 496
AWS CLI 497
Balisage des ressources 499
Principes de base des étiquettes 499
Limites et restrictions liées aux balises 500
Marquer avec les politiques IAM 501
Groupes de facturation 504
Affichage des données de répartition des coûts et d'utilisation 504
Sécurité 507
Sécurité dans AWS IoT 508
Authentification 509

Présentation des certificats X.509 509
Authentification du serveur 509
Authentification client 514
Authentification et autorisation personnalisées 555
Autorisation 586
AWS formation et certification 589
AWS IoT Core politiques 589
Autoriser les appels directs vers les AWS services à l'aide d'un fournisseur AWS IoT Core d'informations d'identification 674
Accès intercompte avec IAM 681
Protection des données 683
Chiffrement des données dans AWS IoT 685
Sûreté des transports dans AWS IoT Core 685
Chiffrement des données 691
Gestion des identités et des accès 692
Public ciblé 693
Authentification avec des identités IAM 693
Gestion des accès à l’aide de politiques 697
Comment AWS IoT fonctionne avec IAM 700
Exemples de politiques basées sur l’identité 733
AWS politiques gérées 738
Résolution des problèmes 753
Journalisation et surveillance 756
Outils de supervision 756
Validation de conformité 758
Résilience 759
Utilisation AWS IoT Core avec des points de terminaison VPC 760
Création de points de terminaison VPC pour le plan de données AWS IoT Core 761
Création de points de terminaison d'un VPC pour le fournisseur d'informations d'identification AWS IoT Core 762
Création d'un point de terminaison d'interface Amazon VPC 763
Configuration d'une zone hébergée privée 764
Contrôle de l'accès aux points de AWS IoT Core terminaison via VPC 766
Limites 768
Dimensionnement des points de terminaison VPC avec AWS IoT Core 769
Utiliser des domaines personnalisés avec des points de terminaison d'un VPC 769

Disponibilité des points de terminaison VPC pour AWS IoT Core 769
Sécurité de l’infrastructure 770
Surveillance de la sécurité 770
Bonnes pratiques de sécurité 771
Protection des connexions MQTT dans AWS IoT 771
Veiller à la synchronisation de l'horloge de votre appareil 774
Valider le certificat de serveur 775
Utiliser une identité unique par appareil 775
Utiliser une seconde Région AWS comme sauvegarde 776
Utiliser la mise en service juste à temps 776
Autorisations pour exécuter des tests AWS IoT Device Advisor 776
Prévention du problème de l'adjoint confus entre services pour l'interface Device Advisor ... 778
AWS formation et certification 779
Moniteur AWS IoT 780
Configuration de la AWS IoT journalisation 781
Configurer le rôle et la stratégie de journalisation 782
Configurez la journalisation par défaut dans AWS IoT (console) 784
Configurer la connexion par défaut AWS IoT (CLI) 786
Configurer la connexion spécifique aux ressources AWS IoT (interface de ligne de commande) 787
Niveaux de journalisation. 791
Surveillez les AWS IoT alarmes et les métriques à l'aide d'Amazon CloudWatch 791
Utilisation de AWS IoT métriques 792
Créez des CloudWatch alarmes 792
Métriques et dimensions 797
Surveiller AWS IoT à l'aide CloudWatch des journaux 821
Afficher AWS IoT les journaux dans la CloudWatch console 822
CloudWatch Enregistre les entrées du AWS IoT journal 823
Téléchargez les journaux côté appareil sur Amazon CloudWatch 861
Comment ça marche 861
Téléchargement de journaux côté appareil à l'aide de AWS IoT règles 862
Enregistrer les appels AWS IoT d'API 872
AWS IoT informations dans CloudTrail 872
Comprendre les entrées du fichier AWS IoT journal 874
Règles 876
Octroi de l’accès 877

Révoquer l'accès au moteur de règles ... 880
Transmettre les autorisations de rôle ... 880
Créer une règle ... 881
Création d'une règle (console) ... 883
Création d'une règle (CLI) ... 884
Gérer une règle ... 889
Marquer une règle ... 889
Afficher une règle ... 890
Suppression d'une règle ... 891
AWS IoT actions liées aux règles ... 891
Apache Kafka ... 894
CloudWatch alarmes ... 907
CloudWatch Journaux ... 909
CloudWatch métriques ... 911
DynamoDB ... 914
Dynamo DBv2 ... 917
Elasticsearch ... 919
HTTP ... 922
IoT Analytics ... 964
AWS IoT Events ... 966
AWS IoT SiteWise ... 968
Firehose ... 974
Kinesis Data Streams ... 977
Lambda ... 979
Emplacement ... 982
OpenSearch ... 986
Republish ... 989
S3 ... 992
Salesforce IoT ... 995
SNS ... 996
SQS ... 998
Step Functions ... 1001
Timestream ... 1003
Résolution des problèmes d'une règle ... 1011
Accédez à des ressources multicomptes ... 1011
Prérequis ... 1012

Configuration entre comptes pour Amazon SQS 1012
Configuration de comptes croisés pour Amazon SNS 1014
Configuration entre comptes pour Amazon S3 1016
Configuration de comptes multiples pour AWS Lambda 1018
Gestion des erreurs (action d'erreur) 1021
Format du message d'action d'erreur 1021
Exemple d'action d'erreur 1023
Basic Ingest 1024
Utilisation de Basic Ingest 1025
AWS IoT Référence SQL 1026
Clause SELECT 1027
Clause FROM 1029
Clause WHERE 1031
Types de données 1031
Opérateurs 1037
Fonctions 1048
Littéraux 1122
Instructions Case 1122
Extensions JSON 1124
Modèles de substitution 1126
Requêtes d'objets imbriqués 1128
Charges utiles binaires 1129
Versions de SQL 1136
Shadows (Ombres) 1139
Utilisation des shadows 1139
Choix d'utilisation de shadows nommés ou non nommés 1140
Accès aux shadows 1140
Utilisation des shadows sur les appareils, dans les applications et dans d'autres services cloud 1141
Ordre des messages 1142
Suppression des messages de shadow 1144
Utilisation des shadows sur les appareils 1145
Initialisation de l'appareil lors de la première connexion à AWS IoT 1146
Traitement des messages lorsque l'appareil est connecté à AWS IoT 1149
Traitement des messages lorsque l'appareil se reconnecte à AWS IoT 1150
Utilisation des shadows dans les applications et les services 1150

Initialisation de l'application ou du service lors de la connexion à AWS IoT 1151
L'état de traitement change lorsque l'application ou le service est connecté à AWS IoT 1151
Détection d'un appareil connecté 1152
Simulation des communications du service Device Shadow 1154
Configuration de la simulation 1154
Initialisation de l'appareil 1154
Envoi d'une mise à jour à partir de l'application 1158
Réponse à une mise à jour sur l'appareil 1161
Observation de la mise à jour dans l'application 1166
Au-delà de la simulation 1168
Interaction avec les shadows 1168
Support du protocole 1169
Demande d'état et génération de rapport d'état 1169
Mise à jour d'un shadow 1169
Récupération d'un document Shadow 1174
Suppression de données shadow 1175
API REST Device Shadow 1178
GetThingShadow 1179
UpdateThingShadow 1180
DeleteThingShadow 1181
ListNamedShadowsForThing 1183
Rubriques MQTT de Device Shadow 1184
/get 1186
/get/accepted 1186
/get/rejected 1187
/update 1188
/update/delta 1190
/update/accepted 1191
/update/documents 1192
/update/rejected 1193
/delete 1194
/delete/accepted 1195
/delete/rejected 1196
Documents du service Device Shadow 1197
Exemples de documents shadow 1197
Propriétés du document 1204

État Delta 1205
Documents shadow de gestion des versions 1207
Jetons clients dans les documents shadow 1207
Propriétés de document shadow vides 1208
Valeurs de tableau dans les documents shadow 1209
Messages d'erreur de Device Shadow 1210
Catalogue de Logiciels 1212
Préparation à l'utilisation du Catalogue de Logiciels 1213
Cycle de vie des versions du package 1213
Conventions de dénomination des versions du package 1215
Version par défaut 1215
Attributs de version 1215
Nomenclature du logiciel 1216
Activation de l'indexation AWS IoT de la flotte 1220
Ombre nommée réservée 1220
Suppression d'un package logiciel 1222
Préparation de la sécurité 1223
Authentification basée sur les ressources 1223
AWS IoT Droits de travail pour déployer des versions de packages 1225
AWS IoT Droits de travail pour mettre à jour l'ombre nommée réservée 1226
AWS IoT Autorisations de téléchargement des jobs depuis Amazon S3 1228
Autorisations de mise à jour de la nomenclature du logiciel pour une version de package .. 1228
Préparation de l'indexation de la flotte 1232
Définir l'`$package`ombre comme source de données 1232
Métriques affichées dans la console 1233
Modèles de requête 1234
Collecte de la distribution des versions de packages via `getBucketsAggregation` 1236
Préparation des AWS IoT emplois 1237
Paramètres de substitution pour les AWS IoT tâches 1237
Préparation du document de travail et de la version du package pour le déploiement 1241
Nommer les packages et les versions lors du déploiement 1245
Cibler les emplois par le biais de groupes d'objets AWS IoT dynamiques 1246
Ombre nommée réservée et versions de package 1246
Désinstallation d'un package logiciel 1247
Premiers pas 1248
Création d'un package et d'une version 1248

Déploiement d'une version de package 1251
Association d'une version de package 1253
Tâches 1255
Accès aux AWS IoT offres d'emploi 1255
AWS IoT Emplois, régions et points de terminaison 1255
Qu'est-ce qu'une opération à distance ? 1256
Avantages de l'utilisation de AWS IoT Device Management Jobs pour les opérations à distance 1256
Qu'est-ce que AWS IoT Jobs ? 1258
Concepts clés relatifs aux tâches 1260
Tâches et états d'exécution des tâches 1264
Gestion des tâches 1269
Signature de code pour les tâches 1270
Document de tâche 1270
Présigné URLs 1270
URL présignée pour le téléchargement de fichiers 1273
URL présignée à l'aide de la gestion des versions d'Amazon S3 1274
Créez et gérez des tâches à l’aide de la console. 1276
Créez et gérez des tâches à l’aide de la CLI. 1279
Modèles de tâche 1291
Modèles personnalisés et AWS gérés 1291
Utiliser des modèles AWS gérés 1292
Créer un modèle de tâche personnalisé 1312
Configuration de la tâche 1321
Comment fonctionnent les configurations de tâches 1321
Configurations supplémentaires 1337
Appareils et tâches 1347
Programmation des appareils pour une utilisation avec Jobs 1350
Flux de travail des appareils 1350
Flux de travail des tâches 1352
Notifications Jobs 1357
AWS IoT opérations de l'API jobs 1365
Types de données de gestion et de contrôle des tâches d’API 1368
Opérations des API MQTT et HTTPS de l’appareil et types de données 1387
Sécuriser les utilisateurs et les appareils pour les tâches 1403
Type de politique requis pour AWS IoT Jobs 1403

Autoriser les utilisateurs de tâches et les services cloud 1404
Autoriser les appareils à utiliser des tâches 1418
AWS IoT Limites d'emplois 1423
Limites relatives à l'exécution des tâches 1423
Limites de tâches actives et simultanées 1425
Commandes 1429
Concepts et statut des commandes 1430
Concepts clés des commandes 1430
États de commande 1432
État de l'exécution de la commande 1433
Flux de travail des commandes 1437
Création et gestion de commandes 1438
Choisissez des cibles et abonnez-vous aux sujets 1439
Démarrez et surveillez les exécutions de commandes 1441
(Facultatif) Activer les notifications pour les événements liés aux commandes 1442
Création et gestion de commandes 1444
Création d'une ressource de commande 1444
Récupérer des informations sur une commande 1448
Répertoriez les commandes dans votre Compte AWS 1451
Mettre à jour une ressource de commande 1452
Dépréciation ou restauration d'une ressource de commande 1455
Supprimer une ressource de commande 1455
Démarrer et surveiller les exécutions de commandes 1457
Lancer l'exécution d'une commande 1458
Mettre à jour le résultat de l'exécution d'une commande 1465
Récupérer l'exécution d'une commande 1471
Affichage des mises à jour des commandes à l'aide du client de test MQTT 1475
Répertoriez les exécutions de commandes dans votre Compte AWS 1477
Supprimer l'exécution d'une commande 1480
Déprécier une ressource de commande 1481
Considérations clés 1481
Déprécier une ressource de commande (console) 1482
Déprécier une ressource de commande (CLI) 1482
Vérifiez l'heure et le statut de la dépréciation 1482
Restaurer une ressource de commande 1483
Tunneling sécurisé 1485

Qu'est-ce que le tunneling sécurisé ? 1485
Concepts de tunneling sécurisés 1486
Comment fonctionne le tunneling sécurisé 1487
Cycle de vie des tunnels sécurisés 1488
Didacticiels sur le tunneling sécurisé 1489
Didacticiels dans cette section 1490
Ouvrez un tunnel et démarrez une session SSH sur un appareil distant 1491
Ouvrez un tunnel pour un appareil distant et utilisez le protocole SSH basé sur un navigateur 1509
Proxy local 1514
Comment utiliser le proxy local 1514
Configuration du proxy local pour les appareils utilisant un proxy Web 1521
Multiplexage et connexions TCP simultanées 1529
Multiplexage de plusieurs flux de données 1530
Utilisation de connexions TCP simultanées 1534
Configuration d'un appareil distant et utilisation de l'agent IoT 1537
Extrait de l'agent IoT 1537
Contrôle de l'accès aux tunnels 1539
Conditions préalables à l'accès au tunnel 1539
Politiques d'accès aux tunnels 1539
Résolution des problèmes de connectivité liés au tunneling sécurisé 1547
Erreur de jeton d'accès client non valide 1548
Erreur de non-concordance du jeton client 1548
Problèmes de connectivité de l'appareil à distance 1550
Mise en service des appareils 1552
Provisionnement d'appareils dans AWS IoT 1553
Approvisionnement de flotte APIs 1555
Mise en service d'appareils qui ne disposent pas de certificats d'appareils à l'aide de la mise en service de flotte 1555
Allocation par revendication 1556
Allocation par utilisateur approuvé 1559
Utilisation des hooks de pré-provisionnement avec l'interface de ligne de commande AWS 1561
Mise en service d'appareils disposant de certificats d'appareils 1565
Mise en service d'un seul objet 1565
Just-in-time approvisionnement 1566
Enregistrement en bloc 1573

Mise en service des modèles 1574
Section Parameters 1574
Section Resources 1575
Exemple de modèle pour l'enregistrement en bloc 1581
Exemple de modèle pour le just-in-time provisionnement (JITP) 1582
Mise en service de flotte 1584
Hooks de mise en service en amont 1588
Entrée du hook de pré-provisionnement 1589
Valeur de retour du hook de pré-provisionnement 1589
Exemple Lambda de hook de mise en service 1590
Signature de certificats autogérée à l'aide d' AWS IoT Core un fournisseur de certificats 1593
Comment fonctionne la signature de certificats autogérés dans le cadre de l'approvisionnement de flottes 1594
Entrée de la fonction Lambda du fournisseur de certificats 1596
Valeur renvoyée par la fonction Lambda du fournisseur de certificats 1596
Exemple de fonction Lambda 1596
Signature de certificats autogérée pour le provisionnement de la flotte 1598
AWS CLI commandes pour le fournisseur de certificats 1600
Création de politiques et de rôles IAM pour un utilisateur installant un appareil 1603
Création d'une politique IAM pour l'utilisateur qui installera un appareil 1603
Création d'un rôle IAM pour l'utilisateur qui installera un appareil 1604
Mettre à jour une politique existante pour autoriser un nouveau modèle 1605
API MQTT de mise en service des appareils 1606
CreateCertificateFromCsr 1607
CreateKeysAndCertificate 1610
RegisterThing 1612
Indexation de la flotte 1615
Gestion des mises à jour des index 1615
Interrogation de l'état de connectivité d'un appareil spécifique 1615
Recherche parmi les sources de données 1615
Interrogation des données agrégées 1616
Surveillance des données agrégées et création d'alarmes à l'aide des indicateurs de flotte 1616
Gestion de l'indexation de la flotte 1616
Indexation d'objets 1616
Indexation du groupe d'objets 1618
Champs gérés 1618

Champs personnalisés 1620
Gérer l'indexation d'objet 1621
Gérer l'indexation du groupe d'objet 1637
État de connectivité de l'appareil 1639
Comment ça marche 1640
Fonctionnalités 1640
Avantages 1640
Prérequis 1641
Exemples 1641
Interrogation des données agrégées 1643
GetStatistics 1643
GetCardinality 1646
GetPercentiles 1647
GetBucketsAggregation 1650
Autorisation 1651
Syntaxe de requête 1651
Fonctionnalités prises en charge 1651
Fonctions non prises en charge 1652
Remarques 1652
Exemples de requêtes sur des objets 1653
Exemples de requêtes sur des groupes d'objets 1657
Indexation des données de localisation 1659
Formats de données pris en charge 1659
Comment indexer les données de localisation 1661
Mettre à jour la configuration de l'indexation des objets 1661
Exemples de géorequêtes 1664
Didacticiel de démarrage 1665
Métriques de la flotte 1670
Didacticiel de démarrage 1670
Gestion des métriques de flotte 1678
Livraison de fichiers basée sur MQTT 1685
Qu'est-ce qu'un stream ? 1685
Gérer un stream 1686
Accordez des autorisations à vos appareils 1687
Connectez vos appareils à AWS IoT 1688
TagResource Usage 1689

Utiliser la livraison de fichiers AWS IoT basée sur MQTT sur les appareils 1690
DescribeStream À utiliser pour obtenir des données de flux 1690
Obtenir des blocs de données à partir d'un fichier de flux 1692
Gestion des erreurs liées à la livraison de AWS IoT fichiers basée sur MQTT 1699
Exemple de cas d'utilisation dans FreeRTOS OTA 1701
Device Advisor 1702
Configuration 1704
Création d’un objet IoT 1704
Créer un rôle IAM à utiliser comme rôle de périphérique 1704
Créez une politique gérée par le client pour qu'un utilisateur IAM puisse utiliser Device Advisor 1708
Création d'un utilisateur IAM pour utiliser Device Advisor 1708
Configurer votre appareil 1711
Premiers pas avec Device Advisor dans la console 1712
Flux de travail Device Advisor 1722
Prérequis 1722
Création d'une définition de suite de tests 1722
Obtenir une définition de suite de tests 1725
Obtenez un point de terminaison de test 1726
Lancer l'exécution d'une suite de test 1726
Exécutez une suite de tests 1727
Arrêter l'exécution d'une suite de tests 1727
Obtenez un rapport de qualification pour une exécution réussie de la suite de tests de qualification 1728
Flux de travail détaillé sur la console Device Advisor 1728
Prérequis 1729
Création d'une définition de suite de tests 1729
Lancer l'exécution d'une suite de test 1736
Arrêter l'exécution d'une suite de tests (facultatif) 1738
Afficher les détails et les journaux d'exécution de la suite de tests 1740
Téléchargez un rapport AWS IoT de qualification 1742
Flux de travail de la console de tests de longue durée 1742
Points de terminaison d’un VPC Device Advisor (AWS PrivateLink) 1751
Considérations relatives aux points de AWS IoT Core Device Advisor terminaison VPC 1752
Créer un point de terminaison de VPC d'interface pour AWS IoT Core Device Advisor 1753
Contrôle de l'accès aux points de AWS IoT Core Device Advisor terminaison via VPC 1753

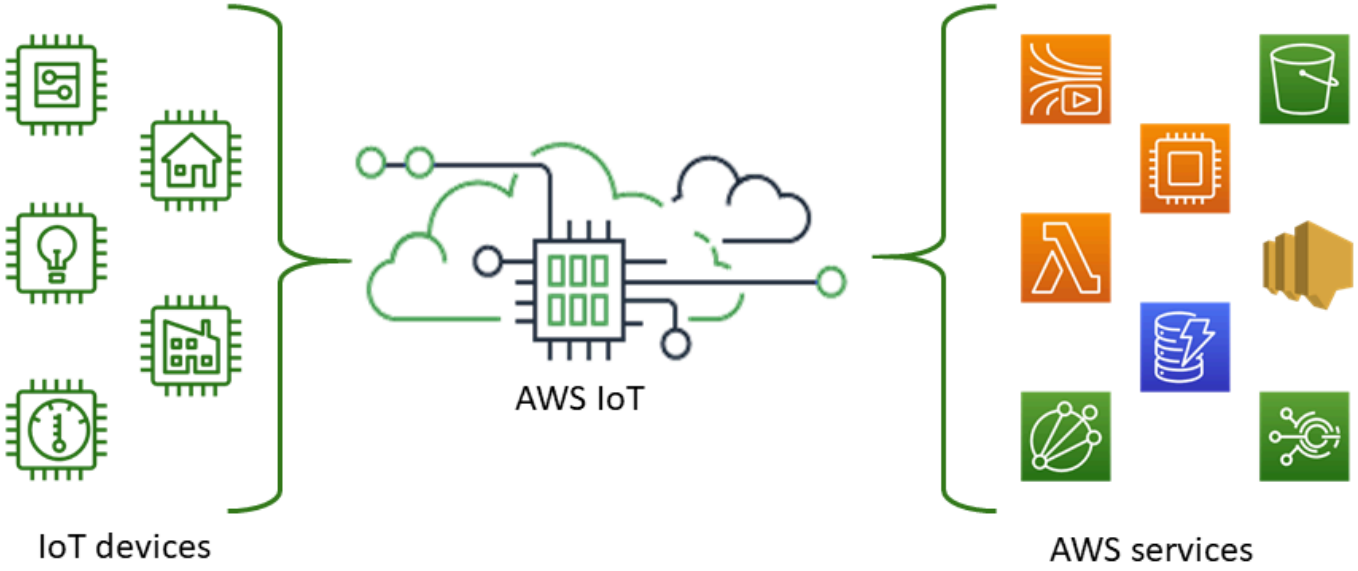
Cas de test Device Advisor 1755
Device Advisor teste des scénarios pour se qualifier pour le programme de qualification des AWS appareils. 1755
TLS 1756
MQTT 1763
Shadow 1778
Exécution d’une tâche 1780
Autorisations et politiques 1782
Tests de longue durée 1783
Emplacement de l'appareil 1801
Types de mesures et solveurs 1802
Comment fonctionne la localisation des AWS IoT Core appareils 1803
Comment utiliser la localisation de AWS IoT Core l'appareil 1804
Résolution de la localisation des appareils IoT 1805
Résolution de la localisation de l'appareil (console) 1806
Résolution de la localisation de l'appareil (API) 1809
Résolution des erreurs lors de la résolution de l'emplacement 1811
Résolution de l'emplacement de l'appareil à l'aide de rubriques MQTT 1812
Format des rubriques MQTT relatives à la localisation des appareils 1812
Rubriques MQTT relatives à la politique de localisation des appareils 1814
Sujets relatifs à la localisation des appareils et charge utile 1815
Solveurs de localisation et charge utile de l'appareil 1820
Solveur basé sur le Wi-Fi 1821
Solveur cellulaire 1822
Solveur de recherche inversée IP 1827
solveur GNSS 1828
Messages d'événements 1830
Comment les messages d'événement sont générés 1830
Politique de réception des messages d'événement 1830
Activez les événements pour AWS IoT 1831
Événements de registre 1836
Événements de l’objet 1836
Événements de types d'objet 1838
Événements de groupe d'objets 1841
Événements Jobs 1847
Événements du cycle de vie 1852

Événements de connexion/déconnexion 1852
Événement d'échec de tentative de connexion 1857
Événements d'abonnement/désabonnement 1858
Résolution des problèmes 1861
AWS IoT Core guide de dépannage 1861
Diagnostic des problèmes de connectivité 1862
Diagnostic des problèmes de règles 1865
Diagnostic des problèmes de shadows 1868
Diagnostic des problèmes liés aux actions Salesforce 1870
Diagnostic des limites de débit 1871
Résolution des problèmes de déconnexion du parc d'appareils 1872
AWS IoT Device Management guide de dépannage 1873
AWS IoT Dépannage des tâches 1873
Résolution des problèmes liés à l'indexation du parc 1878
AWS IoT Résolution des problèmes liés au catalogue des packages logiciels de gestion des appareils 1881
AWS IoT Guide de dépannage de Device Advisor 1889
AWS IoT erreurs 1892
AWS IoT Appareil SDKs, appareil mobile SDKs et client AWS IoT d'appareil 1894
AWS IoT Device SDKs 1894
AWS IoT SDK de périphérique pour Embedded C 1896
AWS Applications mobiles SDKs 1897
AWS IoT Client de l'appareil 1898
SDKs Versions antérieures AWS IoT de l'appareil 1899
Exemples de code 1900
Principes de base 1906
Bonjour AWS IoT 1907
Principes de base 1912
Actions 1968
AWS IoTQuotas 2032
Tarification d'AWS IoT Core 2033
..... mmxxxiv

Qu'est-ce que c'est AWS IoT ?

AWS IoT fournit les services cloud qui connectent vos appareils IoT à d'autres appareils et services AWS cloud. AWS IoT fournit un logiciel qui peut vous aider à intégrer vos appareils IoT dans des solutions AWS IoT basées sur ces appareils. Si vos appareils peuvent se connecter à AWS IoT, AWS IoT vous pouvez les connecter aux services cloud qui les AWS fournissent.

Pour une introduction pratique à AWS IoT, rendez-vous surDidacticiel de démarrage.



Avec AWS IoT, vous pouvez sélectionner les up-to-date technologies les plus appropriées pour votre solution. Pour vous aider à gérer et à soutenir vos appareils IoT sur le terrain, AWS IoT Core prend en charge les protocoles suivants :

- MQTT (file d'attente des messages et transport de télémétrie)
- MQTT sur WSS (Websockets Secure)
- HTTPS (Hypertext Transfer Protocol - Secure)
- LoRaWAN (réseau étendu à longue portée)

Le courtier de AWS IoT Core messages prend en charge les appareils et les clients qui utilisent les protocoles MQTT et MQTT sur WSS pour publier des messages et s'y abonner. Il prend également en charge les appareils et les clients qui utilisent le protocole HTTPS pour publier des messages.

AWS IoT Core for LoRa WAN vous permet de connecter et de gérer des appareils LoRa WAN (réseau étendu longue portée à faible consommation) et à les gérer. AWS IoT Core for LoRa WAN vous évite de devoir développer et exploiter un serveur réseau LoRa WAN (LNS).

Si vous n'avez pas besoin de AWS IoT fonctionnalités telles que les communications entre appareils, les règles ou les tâches, consultez AWS Messagerie pour obtenir des informations sur d'autres services de AWS IoT messagerie susceptibles de mieux répondre à vos besoins.

Comment vos appareils et applications accèdent AWS IoT

AWS IoT fournit les interfaces suivantes pour AWS IoT tutoriels :

- AWS IoT Appareil SDKs : créez sur vos appareils des applications qui envoient et reçoivent des AWS IoT messages. Pour de plus amples informations, veuillez consulter AWS IoT Appareil SDKs, appareil mobile SDKs et client AWS IoT d'appareil.
- AWS IoT Core pour le LoRa WAN —Connectez et gérez vos appareils et passerelles WAN (LoRaWAN) à longue portée en utilisant AWS IoT Core for LoRa WAN.
- AWS Command Line Interface (AWS CLI) —Exécutez des commandes pour AWS IoT Windows, macOS et Linux. Ces commandes vous permettent de créer et de gérer des objets, des certificats, des règles, des tâches et des politiques. Consultez le AWS Command Line Interface Guide de l'utilisateur pour démarrer. Pour plus d'informations sur les commandes pour AWS IoT, consultez iot dans la référence des AWS CLI commandes.
- AWS IoT API —Créez vos applications IoT à l'aide de requêtes HTTP ou HTTPS. Ces actions API vous permettent de créer et de gérer par programme des objets, des certificats, des règles et des politiques. Pour plus d'informations sur les actions d'API pour AWS IoT, consultez la section Actions de la référence AWS IoT d'API.
- AWS SDKs—Créez vos applications IoT en utilisant un langage APIs spécifique. Ils SDKs encapsulent l' HTTP/HTTPS API et vous permettent de programmer dans n'importe laquelle des langues prises en charge. Pour plus d'informations, reportez-vous à la section AWS SDKs et Outils.

Vous pouvez également y accéder AWS IoT via la AWS IoT console, qui fournit une interface utilisateur graphique (GUI) grâce à laquelle vous pouvez configurer et gérer les objets, les certificats, les règles, les tâches, les politiques et les autres éléments de vos solutions IoT.

Ce que AWS IoT je peux faire

Cette rubrique décrit certaines des solutions de AWS IoT support dont vous pourriez avoir besoin.

L'IoT dans l'industrie

Voici quelques exemples de AWS IoT solutions pour des cas d'utilisation industrielle qui appliquent les technologies de l'IoT pour améliorer les performances et la productivité des processus industriels.

Solutions pour les cas d'utilisation industriels

- Utilisation AWS IoT pour créer des modèles de qualité prédictifs dans les opérations industrielles

  Découvrez AWS IoT comment collecter et analyser les données des opérations industrielles pour créer des modèles de qualité prédictifs. En savoir plus
- Utilisation AWS IoT pour soutenir la maintenance prédictive dans les opérations industrielles

  Découvrez comment AWS IoT vous pouvez vous aider à planifier la maintenance préventive afin de réduire les temps d'arrêt imprévus. En savoir plus

L'IoT dans la domotique

Voici quelques exemples de AWS IoT solutions pour les applications domotiques qui appliquent les technologies IoT pour créer des applications IoT évolutives qui automatisent les activités domestiques à l'aide d'appareils domestiques connectés.

Solutions pour la domotique

- Utilisation AWS IoT dans votre maison connectée

  Découvrez comment nous AWS IoT pouvons fournir des solutions domotiques intégrées.

- Utilisé AWS IoT pour assurer la sécurité et la surveillance de la maison

  Découvrez AWS IoT comment appliquer l'apprentissage automatique et l'informatique de pointe à votre solution domotique.

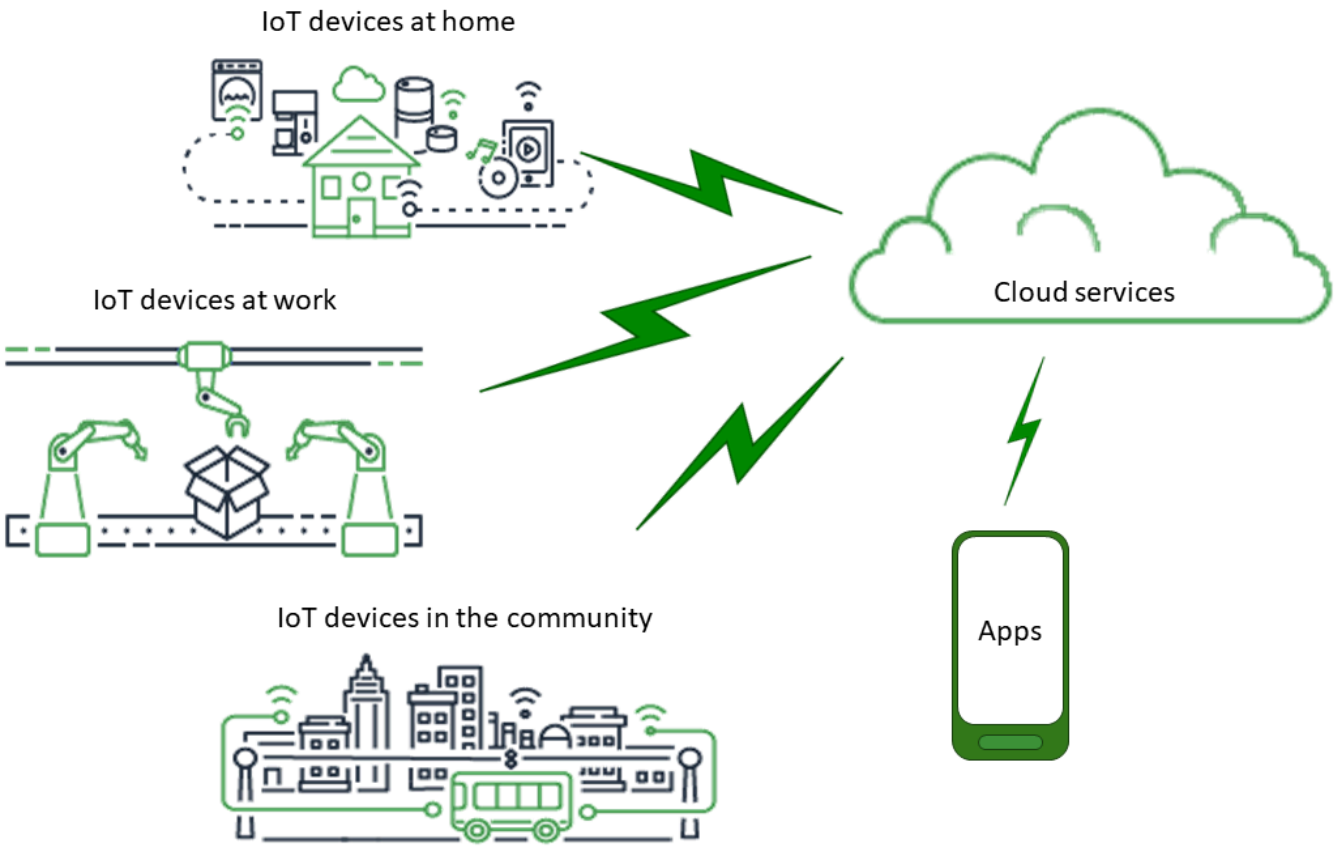
Pour obtenir une liste des solutions pour les cas d'utilisation industriels, grand public et commerciaux, consultez le AWS IoT référentiel de solutions.

Comment AWS IoT fonctionne

AWS IoT fournit des services cloud et une assistance pour les appareils que vous pouvez utiliser pour mettre en œuvre des solutions IoT. AWS fournit de nombreux services cloud pour prendre en charge les applications basées sur l'IoT. Pour vous aider à comprendre par où commencer, cette section fournit un schéma et une définition des concepts essentiels pour vous présenter l'univers de l'IoT.

L'univers de l'IoT Side

En général, l'Internet des objets (IoT) comprend les composants clés illustrés dans ce schéma.



Applications

Les applications permettent aux utilisateurs finaux d'accéder aux appareils IoT et aux fonctionnalités fournies par les services cloud auxquels ces appareils sont connectés.

Les services cloud

Les services cloud sont des services de stockage et de traitement de données distribués à grande échelle connectés à Internet. En voici quelques exemples :

- Services de connexion et de gestion de l'IoT

  AWS IoT est un exemple de service de connexion et de gestion de l'IoT.
- Services informatiques, tels qu'Amazon Elastic Compute Cloud et AWS Lambda
- Services de base de données, tels qu'Amazon DynamoDB

Communications

Les appareils communiquent avec les services cloud à l'aide de diverses technologies et protocoles. En voici quelques exemples :

- Wi-Fi/Internet haut débit
- Données cellulaires à haut débit
- Données cellulaires à bande étroite
- Réseau étendu (LoRaWAN) longue portée
- Communications RF propriétaires

Appareils

Un appareil est un type de matériel qui gère les interfaces et les communications. Les appareils sont généralement situés à proximité des interfaces réelles qu'ils surveillent et contrôlent. Les appareils peuvent inclure des ressources informatiques et de stockage, telles que des microcontrôleurs, des processeurs, de la mémoire. En voici quelques exemples :

- Raspberry Pi
- Arduino
- Assistants d'interface vocale
- LoRaWAN et appareils
- Appareils Amazon Sidewalk
- Appareils IoT personnalisés

Interfaces

Une interface est un composant qui connecte un appareil au monde physique.

- Interfaces utilisateur

  Des composants qui permettent aux appareils et aux utilisateurs de communiquer entre eux.

  - Interfaces d'entrée

    Permettre à un utilisateur de communiquer avec un appareil

Exemples : clavier, bouton

- Interfaces de sortie

  Permettre à un appareil de communiquer avec un utilisateur

  Exemples : affichage alphanumérique, affichage graphique, voyant lumineux, sonnette d'alarme

- Sensors

  Composants d'entrée qui mesurent ou détectent quelque chose dans le monde extérieur d'une manière compréhensible par un appareil. En voici quelques exemples :

  - Capteur de température (convertit la température en signal analogique ou numérique)
  - Capteur d'humidité (convertit l'humidité relative en signal analogique ou numérique)
  - Convertisseur analogique-numérique (convertit une tension analogique en valeur numérique)
  - Unité de mesure de distance à ultrasons (convertit une distance en valeur numérique)
  - Capteur optique (convertit un niveau de lumière en valeur numérique)
  - Appareil photo (convertit les données d'image en données numériques)

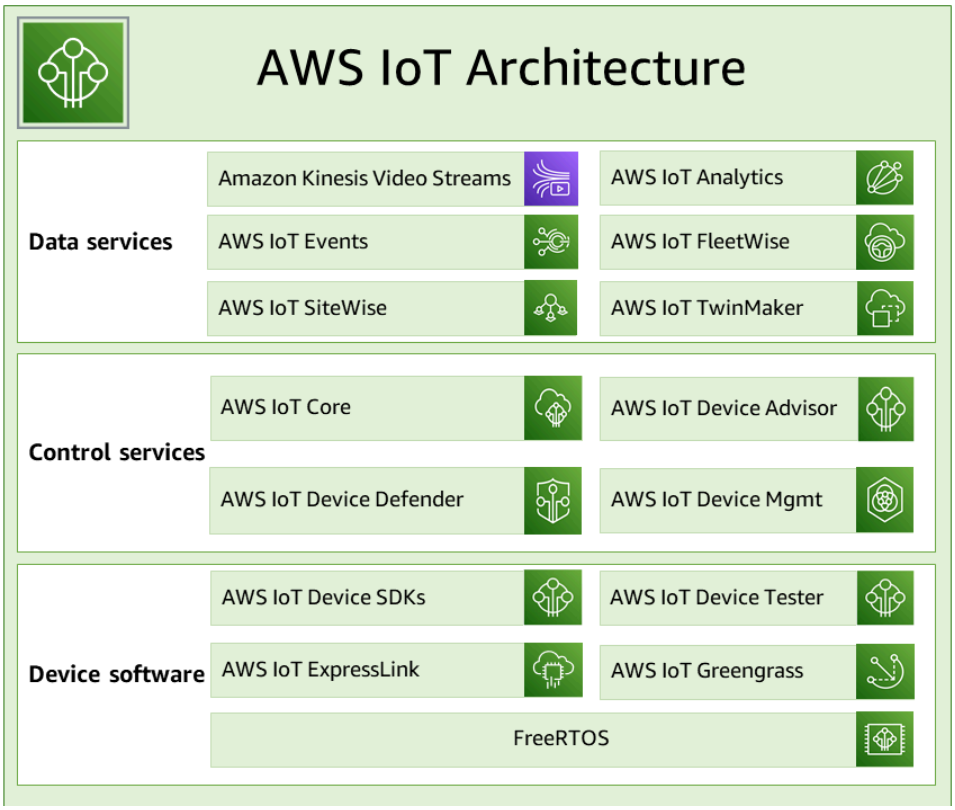
- Actuators

  Composants de sortie que l'appareil peut utiliser pour contrôler quelque chose dans le monde extérieur. En voici quelques exemples :

  - Moteurs pas à pas (convertissent les signaux électriques en mouvement)
  - Relais (contrôlant les tensions et courants électriques élevés)

AWS IoT aperçu des services

Dans l'univers de l'IoT, AWS IoT fournit les services qui prennent en charge les appareils qui interagissent avec le monde et les données qui circulent entre eux et AWS IoT. AWS IoT est composé des services présentés dans cette illustration pour soutenir votre solution IoT.



AWS IoT logiciel de l'appareil

AWS IoT fournit ce logiciel pour prendre en charge vos appareils IoT.

AWS IoT Device SDKs

L'AWS IoT appareil et le mobile vous SDKs aident à connecter efficacement vos appareils à AWS IoT. The AWS IoT Device et Mobile SDKs incluent des bibliothèques open source, des guides de développement avec des exemples et des guides de portage afin que vous puissiez créer des produits ou des solutions IoT innovants sur les plateformes matérielles de votre choix.

AWS IoT Device Tester

AWS IoT Device Testerpour FreeRTOS AWS IoT Greengrass et est un outil d'automatisation des tests pour les microcontrôleurs. AWS IoT Device Tester teste votre appareil pour déterminer s'il exécutera FreeRTOS AWS IoT Greengrass ou interagira avec les services. AWS IoT

AWS IoT ExpressLink

AWS IoT ExpressLink alimente une gamme de modules matériels développés et proposés par les AWS partenaires. Les modules de connectivité incluent un logiciel AWS validé, ce qui vous permet de connecter plus rapidement et plus facilement des appareils au cloud en toute

sécurité et de les intégrer parfaitement à une gamme de AWS services. Pour plus d'informations, consultez la page de AWS IoT ExpressLinkprésentation ou le guide du AWS IoT ExpressLink programmeur.

AWS IoT Greengrass

AWS IoT Greengrasss'étend AWS IoT aux appareils périphériques afin qu'ils puissent agir localement sur les données qu'ils génèrent, exécuter des prédictions basées sur des modèles d'apprentissage automatique, et filtrer et agréger les données des appareils. AWS IoT Greengrass permet à vos appareils de collecter et d'analyser des données plus près de l'endroit où elles sont générées, de réagir de manière autonome aux événements locaux et de communiquer en toute sécurité avec les autres appareils du réseau local. Vous pouvez l'utiliser AWS IoT Greengrass pour créer des applications de périphérie à l'aide de modules logiciels prédéfinis, appelés composants, qui peuvent connecter vos appareils de périphérie à des AWS services ou à des services tiers.

FreeRTOS

FreeRTOS est un système d'exploitation open source en temps réel pour les microcontrôleurs qui vous permet d'inclure de petits appareils périphériques à faible consommation d'énergie dans votre solution IoT. FreeRTOS inclut un noyau et un ensemble croissant de bibliothèques logicielles qui prennent en charge de nombreuses applications. Les systèmes FreeRTOS peuvent connecter en toute sécurité vos petits appareils à faible consommation à AWS IoT et prendre en charge le fonctionnement d'appareils plus puissant AWS IoT Greengrass.

AWS IoT services de contrôle

Connectez-vous aux AWS IoT services suivants pour gérer les appareils de votre solution IoT.

AWS IoT Core

AWS IoT Coreest un service cloud géré qui permet aux appareils connectés d'interagir en toute sécurité avec des applications cloud et d'autres appareils. AWS IoT Core peut prendre en charge de nombreux appareils et messages, et il peut traiter et acheminer ces messages vers des AWS IoT terminaux et d'autres appareils. Grâce à cela AWS IoT Core, vos applications peuvent interagir avec tous vos appareils même lorsqu'ils ne sont pas connectés.

AWS IoT Core Conseiller en matière d'appareils

AWS IoT Core Device Advisor est une fonctionnalité de test entièrement gérée basée sur le cloud qui permet de valider les appareils IoT lors du développement du logiciel des appareils. Device

Advisor propose des tests prédéfinis que vous pouvez utiliser pour valider les appareils IoT afin de garantir une connectivité fiable et sécurisée AWS IoT Core, avant de les déployer en production.

AWS IoT Défenseur de l'appareil

AWS IoT Device Defender vous aide à sécuriser votre parc d'appareils IoT. AWS IoT Device Defender analyse en permanence vos configurations IoT pour s'assurer qu'elles ne s'écartent pas des meilleures pratiques en matière de sécurité. AWS IoT Device Defender envoie une alerte lorsqu'il détecte des lacunes dans votre configuration IoT susceptibles de créer un risque de sécurité, comme le partage de certificats d'identité entre plusieurs appareils ou la tentative de connexion d'un appareil dont le certificat d'identité a été révoqué. AWS IoT Core

AWS IoT Gestion des appareils

AWS IoT Les services de gestion des appareils vous aident à suivre, surveiller et gérer la pléthore d'appareils connectés qui constituent vos flottes d'appareils. AWS IoT Les services de gestion des appareils vous aident à garantir que vos appareils IoT fonctionnent correctement et en toute sécurité après leur déploiement. Ils fournissent également un tunneling sécurisé pour accéder à vos appareils, surveiller leur état de santé, détecter et résoudre les problèmes à distance, ainsi que des services permettant de gérer les mises à jour du logiciel et du microprogramme des appareils.

AWS IoT services de données

Analysez les données des appareils de votre solution IoT et prenez les mesures appropriées en utilisant les AWS IoT services suivants.

Amazon Kinesis Video Streams

Amazon Kinesis Video Streams vous permet de diffuser des vidéos en direct depuis des appareils vers AWS le cloud, où elles sont stockées, cryptées et indexées de manière durable, ce qui vous permet d'accéder à vos données via. easy-to-use APIs Vous pouvez utiliser Amazon Kinesis Video Streams pour capturer des quantités massives de données vidéo en direct provenant de millions de sources, notamment des smartphones, des caméras de sécurité, des webcams, des caméras embarquées dans des voitures, des drones et d'autres sources. Amazon Kinesis Video Streams vous permet de visionner des vidéos en direct et à la demande, et de créer rapidement des applications qui tirent parti de la vision par ordinateur et de l'analyse vidéo grâce à l'intégration à Amazon Rekognition Video et aux bibliothèques pour les frameworks de machine learning. Vous pouvez également envoyer données non vidéo sérialisées, par exemple des

données audio, de l'imagerie thermique, de données de profondeur, des données RADAR, et bien plus encore.

Amazon Kinesis Video Streams

Amazon Kinesis Video Streams with WebRTC fournit une implémentation WebRTC conforme aux normes en tant que fonctionnalité entièrement gérée. Vous pouvez utiliser Amazon Kinesis Video Streams avec WebRTC pour diffuser du contenu multimédia en direct en toute sécurité ou effectuer une interaction audio ou vidéo bidirectionnelle entre n'importe quel appareil photo IoT et des lecteurs mobiles ou Web compatibles WebRTC. En tant que fonctionnalité entièrement gérée, vous n'avez pas besoin de créer, d'exploiter ou de faire évoluer une infrastructure cloud liée au WebRTC, telle que des serveurs de signalisation ou de relais multimédia pour diffuser du contenu multimédia en toute sécurité sur des applications et des appareils. En utilisant Amazon Kinesis Video Streams avec WebRTC, vous pouvez facilement créer des applications pour le streaming multimédia peer-to-peer en direct ou pour l'interactivité audio ou vidéo en temps réel entre appareils photo IoT, navigateurs Web et appareils mobiles pour de nombreux cas d'utilisation.

AWS IoT Analytique

AWS IoT L'analytique vous permet d'exécuter et de rendre opérationnelles efficacement des analyses sophistiquées sur d'énormes volumes de données IoT non structurées. AWS IoT L'analytique automatise chaque étape difficile requise pour analyser les données des appareils IoT. AWS IoT L'analytique filtre, transforme et enrichit les données IoT avant de les stocker dans un magasin de données chronologiques à des fins d'analyse. Vous pouvez analyser vos données en exécutant des requêtes ponctuelles ou planifiées à l'aide du moteur de requêtes SQL intégré ou de l'apprentissage automatique.

AWS IoT Événements

AWS IoT Events détecte les événements provenant des capteurs et des applications IoT et y répond. Les événements sont des modèles de données qui identifient des circonstances plus complexes que prévu, comme les détecteurs de mouvement utilisant des signaux de mouvement pour activer les lumières et les caméras de sécurité. AWS IoT Events surveille en permanence les données provenant de plusieurs capteurs et applications IoT, et s'intègre à d'autres services AWS IoT Core, tels que l'IoT SiteWise, DynamoDB, etc. pour permettre une détection précoce et des informations uniques.

AWS IoT FleetWise

AWS IoT FleetWiseest un service géré que vous pouvez utiliser pour collecter et transférer les données des véhicules vers le cloud en temps quasi réel. Avec AWS IoT FleetWise, vous pouvez

facilement collecter et organiser les données provenant de véhicules utilisant différents protocoles et formats de données. AWS IoT FleetWise permet de transformer les messages de bas niveau en valeurs lisibles par l'homme et de normaliser le format des données dans le cloud pour les analyses de données. Vous pouvez également définir des schémas de collecte de données pour contrôler les données à collecter dans les véhicules et le moment de les transférer vers le cloud.

AWS IoT SiteWise

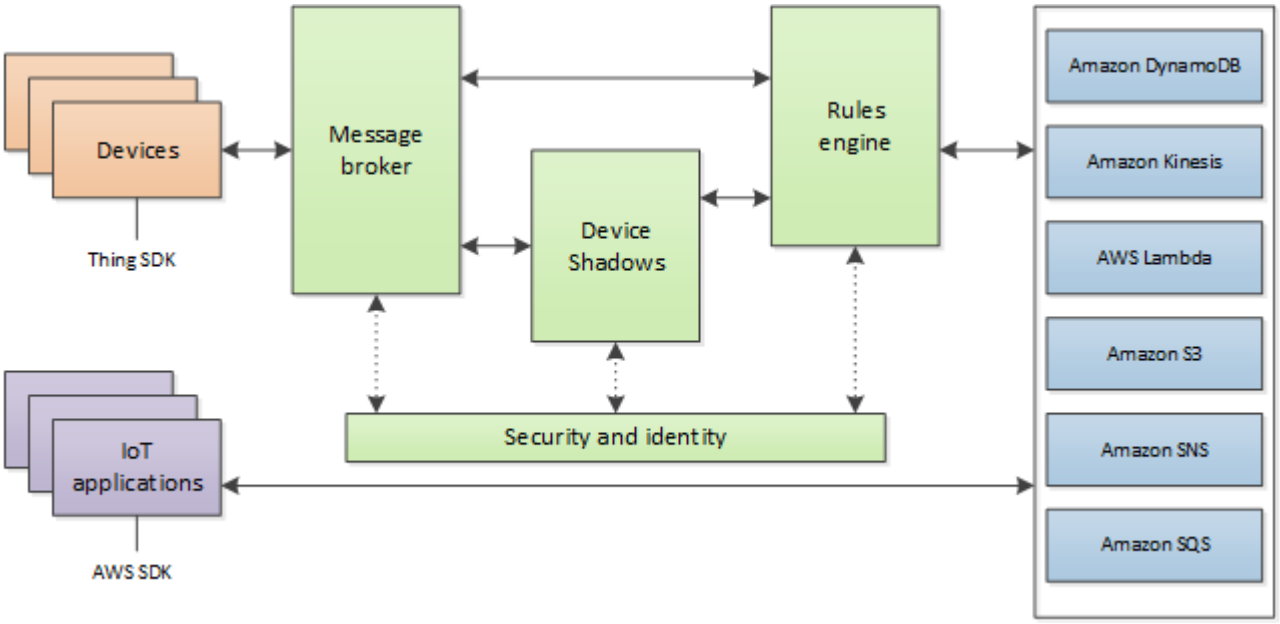
AWS IoT SiteWisecollecte, stocke, organise et surveille les données transmises par les équipements industriels par le biais de messages MQTT ou APIs à grande échelle en fournissant un logiciel qui s'exécute sur une passerelle dans vos installations. La passerelle se connecte en toute sécurité à vos serveurs de données sur site et automatise le processus de collecte et d'organisation des données et leur envoi vers le AWS cloud.

AWS IoT TwinMaker

AWS IoT TwinMakerconstruit des jumeaux numériques opérationnels de systèmes physiques et numériques. AWS IoT TwinMaker crée des visualisations numériques à l'aide de mesures et d'analyses issues de divers capteurs, caméras et applications d'entreprise du monde réel pour vous aider à suivre votre usine physique, votre bâtiment ou votre installation industrielle. Vous pouvez utiliser des données réelles pour surveiller les opérations, diagnostiquer et corriger les erreurs, et optimiser les opérations.

AWS IoT Core services

AWS IoT Core fournit les services qui connectent vos appareils IoT au AWS cloud afin que d'autres services et applications cloud puissent interagir avec vos appareils connectés à Internet.



La section suivante décrit chacun des AWS IoT Core services présentés dans l'illustration.

AWS IoT Core services de messagerie

Les services de AWS IoT Core connectivité fournissent une communication sécurisée avec les appareils IoT et gèrent les messages qui passent entre eux et AWS IoT.

Passerelle pour les appareils

Permet aux appareils de communiquer de manière efficace et en toute sécurité avec AWS IoT. La communication entre appareils est sécurisée par des protocoles sécurisés qui utilisent des certificats X.509.

Agent de messages

Fournit un mécanisme sécurisé permettant aux appareils et aux AWS IoT applications de publier et de recevoir des messages les uns des autres. Vous pouvez utiliser le protocole MQTT directement ou MQTT over WebSocket pour publier et vous abonner. Pour plus d'informations sur les protocoles et les ports AWS IoT pris en charge, consultez the section called “Protocoles de communication des appareils”. Les appareils et les clients peuvent également utiliser l'interface HTTP REST pour publier des données sur le courtier de messages.

Le courtier de messages distribue les données des appareils aux appareils qui y sont abonnés et à d'autres AWS IoT Core services, tels que le service Device Shadow et le moteur de règles.

AWS IoT Core pour LoRa WAN

AWS IoT Core for LoRa WAN permet de configurer un réseau LoRa WAN privé en connectant vos appareils LoRa WAN et vos passerelles AWS sans avoir à développer et à exploiter un serveur de réseau LoRa WAN (LNS). Les messages reçus des périphériques LoRa WAN sont envoyés au moteur de règles où ils peuvent être formatés et envoyés à d'autres AWS IoT services.

Moteur de règles

Le moteur de règles connecte les données du courtier de messages à d'autres AWS IoT services à des fins de stockage et de traitement supplémentaire. Par exemple, vous pouvez insérer, mettre à jour ou interroger une table DynamoDB ou appeler une fonction Lambda en fonction d'une expression que vous avez définie dans le moteur de règles. Vous pouvez utiliser un langage SQL pour sélectionner des données dans les messages, puis traiter et envoyer les données à d'autres services, tels que Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, et AWS Lambda. Vous pouvez également créer des règles qui republient les messages vers le courtier en messages, puis vers d'autres abonnés. Pour de plus amples informations, veuillez consulter Règles pour AWS IoT.

AWS IoT Core services de contrôle

Les services AWS IoT Core de contrôle fournissent des fonctionnalités de sécurité, de gestion et d'enregistrement des appareils.

Service d'authentification personnalisé

Vous pouvez définir des autorisateurs personnalisés qui vous permettent de gérer votre propre stratégie d'authentification et d'autorisation à l'aide d'un service d'authentification personnalisé et d'une fonction Lambda. Les autorisateurs personnalisés permettent d' AWS IoT authentifier vos appareils et d'autoriser les opérations à l'aide de stratégies d'authentification et d'autorisation par jeton porteur.

Les autorisateurs personnalisés peuvent mettre en œuvre différentes stratégies d'authentification, par exemple la vérification du jeton Web JSON ou l'appel du OAuth fournisseur. Ils doivent renvoyer les documents de politique utilisés par la passerelle du périphérique pour autoriser les opérations MQTT. Pour de plus amples informations, veuillez consulter Authentification et autorisation personnalisées.

Service de mise en service d'appareils

Permet de mettre en service des appareils à l'aide d'un modèle qui décrit les ressources requises par chaque appareil : un objet, un certificat et une ou plusieurs stratégies. Un objet est une entrée dans le registre qui contient des attributs décrivant un appareil. Les appareils utilisent des certificats pour s'authentifier. AWS IoT Les stratégies déterminent les opérations qu'un appareil peut effectuer dans AWS IoT.

Les modèles contiennent des variables remplacées par des valeurs dans un dictionnaire (map). Vous pouvez utiliser le même modèle pour mettre en service plusieurs appareils en transmettant différentes valeurs pour les variables du modèle dans le dictionnaire. Pour de plus amples informations, veuillez consulter Mise en service des appareils.

Registre de groupe

Les groupes vous permettent de gérer plusieurs appareils simultanément en les classant dans des groupes. Les groupes peuvent également contenir des groupes—vous pouvez construire une hiérarchie de groupes. Toute action que vous effectuez sur un groupe parent s'appliquera à ses groupes enfants. La même action s'applique également à tous les appareils du groupe parent et à tous les appareils des groupes enfants. Les autorisations accordées à un groupe s'appliquent à tous les appareils du groupe et de tous ses groupes enfants. Pour de plus amples informations, veuillez consulter Gestion des appareils avec AWS IoT.

Service Jobs

Vous permet de définir un ensemble d'opérations distantes qui sont envoyées vers un ou plusieurs appareils connectés à AWS IoT. Par exemple, vous pouvez définir une tâche qui ordonne à un ensemble d'appareils de télécharger et d'installer les mises à jour d'une application ou d'un microprogramme, de redémarrer, de procéder à une rotation des certificats ou d'exécuter des opérations de dépannage à distance.

Pour créer une tâche, vous devez préciser une description des opérations à distance à effectuer et une liste des cibles qui doivent les effectuer. Les cibles peuvent être des appareils individuels, des groupes ou les deux. Pour de plus amples informations, veuillez consulter AWS IoT Emplois.

Registre

Organise les ressources associées à chaque appareil dans le AWS cloud. Vous enregistrez vos appareils et associez jusqu'à trois attributs personnalisés à chacun d'entre eux. Vous pouvez également associer des certificats et un client MQTT IDs à chaque appareil pour améliorer votre capacité à les gérer et à les dépanner. Pour de plus amples informations, veuillez consulter Gestion des appareils avec AWS IoT.

Service de sécurité et d'identité

Fournit une responsabilité partagée en matière de sécurité dans le AWS cloud. Vos appareils doivent conserver leurs informations d'identification pour envoyer en toute sécurité des données au courtier de messages. L'agent de messages et les règles de message utilisent les fonctions de AWS sécurité pour envoyer des données en toute sécurité vers des appareils ou autres AWS services. Pour de plus amples informations, veuillez consulter Authentification.

AWS IoT Core services de données

Les services de AWS IoT Core données aident vos solutions IoT à fournir une expérience applicative fiable, même avec des appareils qui ne sont pas toujours connectés.

Shadow d'appareil

Document JSON utilisé pour stocker et récupérer des informations d'état actualisées concernant un appareil.

Service Device Shadow

Le service Device Shadow maintient l'état d'un appareil afin que les applications puissent communiquer avec celui-ci, que celui-ci soit en ligne ou non. Lorsqu'un appareil est hors ligne, le service Device Shadow gère ses données pour les applications connectées. Lorsque l'appareil se reconnecte, il synchronise son état avec celui de son ombre dans le service Device Shadow. Vos appareils peuvent également publier leur état actuel dans une zone d'ombre à l'intention d'applications ou d'autres appareils qui ne sont pas connectés en permanence. Pour de plus amples informations, veuillez consulter AWS IoT Service Device Shadow.

AWS IoT Core service d'assistance

Intégration d'Amazon Sidewalk pour AWS IoT Core

Amazon Sidewalk est un réseau partagé qui améliore les options de connectivité afin d'aider les appareils à mieux fonctionner ensemble. Amazon Sidewalk prend en charge un large éventail d'appareils clients, tels que ceux qui permettent de localiser les animaux de compagnie ou les objets de valeur, ceux qui fournissent un contrôle intelligent de la sécurité et de l'éclairage pour les maisons, et ceux qui fournissent des diagnostics à distance pour les appareils et les outils. Amazon Sidewalk Integration for AWS IoT Core permet aux fabricants d'appareils d'ajouter leur parc d'appareils Sidewalk au AWS IoT cloud.

Pour de plus amples informations, veuillez consulter AWS IoT Core pour Amazon Sidewalk.

En savoir plus sur AWS IoT

Cette rubrique vous permet de vous familiariser avec le monde de AWS IoT. Vous pouvez obtenir des informations générales sur la manière dont les solutions IoT sont appliquées dans différents cas d'utilisation, des ressources de formation, des liens vers les réseaux sociaux AWS IoT et tous les autres AWS services, ainsi qu'une liste des services et protocoles de communication AWS IoT utilisés.

Ressources de formation pour AWS IoT

Nous proposons ces cours de formation pour vous aider à mieux les connaître AWS IoT et à les appliquer à la conception de votre solution.

- Présentation de AWS IoT

  Un aperçu vidéo AWS IoT de ses principaux services.
- Approfondissement de l' AWS IoT authentification et de l'autorisation

  Un cours avancé qui explore les concepts d' AWS IoT authentification et d'autorisation. Vous apprendrez comment authentifier et autoriser les clients à accéder au plan de AWS IoT contrôle et au plan de données. APIs
- Série de la Fondation pour l'Internet des objets

  Un parcours d'apprentissage composé de modules d'apprentissage en ligne sur les différentes technologies et fonctionnalités de l'IoT.

AWS IoT ressources et guides

Il s'agit de ressources techniques approfondies sur des aspects spécifiques de AWS IoT.

- IoT Lens — Cadre AWS IoT Well-Architected

  Un document qui décrit les meilleures pratiques pour l'architecture de vos applications IoT sur AWS.
- Conception de sujets MQTT pour AWS IoT Core

Livre blanc qui décrit les meilleures pratiques pour concevoir des sujets MQTT dans MQTT AWS IoT Core et en tirer parti AWS IoT Core .

- Résumé et introduction

  Document PDF qui décrit les différentes méthodes AWS IoT permettant de provisionner de grands parcs d'appareils.

- AWS IoT Core Device Advisor

  AWS IoT Core Device Advisor propose des tests prédéfinis que vous pouvez utiliser pour valider les meilleures pratiques de connectivité fiables et sécurisées avec les appareils IoT AWS IoT Core, avant de les déployer en production.

- AWS IoT Ressources

  Des ressources spécifiques à l'IoT, telles que des guides techniques, des architectures de référence, des livres électroniques et des articles de blog sélectionnés, sont présentées dans un index consultable.

- IoT Atlas

  Des aperçus sur la manière de résoudre les problèmes de conception courants liés à l'IoT. L'Atlas de l'IoT fournit une analyse approfondie des défis de conception que vous êtes susceptible de rencontrer lors du développement de votre solution IoT.

- AWS Livres blancs et guides

  Notre collection actuelle de livres blancs, de guides et d'autres AWS IoT AWS technologies.

AWS IoT sur les réseaux sociaux

Ces réseaux sociaux fournissent des informations sur AWS IoT des AWS sujets connexes.

- L'Internet des objets en ligne AWS IoT — Blog officiel
- AWS IoT vidéos de la chaîne Amazon Web Services sur YouTube

Ces comptes de réseaux sociaux couvrent tous les AWS services, y compris AWS IoT

- La chaîne Amazon Web Services sur YouTube
- Amazon Web Services sur Twitter

- Amazon Web Services sur Facebook
- Amazon Web Services sur Instagram
- Amazon Web Services sur LinkedIn

AWS services utilisés par le moteur de AWS IoT Core règles

Le moteur de AWS IoT Core règles peut se connecter à ces AWS services.

- Amazon DynamoDB

  Amazon DynamoDB est un service de base de données NoSQL évolutif qui offre des performances de base de données rapides et prévisibles.

- Amazon Kinesis

  Amazon Kinesis facilite la collecte, le traitement et l'analyse des données diffusées en temps réel afin que vous puissiez obtenir des informations pertinentes et réagir rapidement aux nouvelles informations. Amazon Kinesis peut ingérer des données en temps réel telles que des données vidéo, audio, des journaux d'applications, des flux de clics sur des sites Web et des données de télémétrie IoT à des fins d'apprentissage automatique, d'analyse et d'autres applications.

- AWS Lambda

  AWS Lambda vous permet d'exécuter du code sans provisionner ni gérer de serveurs. Vous pouvez configurer votre code pour qu'il se déclenche automatiquement à partir de AWS IoT données et d'événements ou l'appeler directement depuis une application Web ou mobile.

- Amazon Simple Storage Service

  Amazon Simple Storage Service (Amazon S3) peut stocker et récupérer n'importe quel volume de données à tout moment, où que vous soyez sur le Web. AWS IoT les règles peuvent envoyer des données à Amazon S3 à des fins de stockage.

- Amazon Simple Notification Service

  Amazon Simple Notification Service (Amazon SNS) est un service web qui permet aux applications, aux utilisateurs finaux et aux appareils d'envoyer et de recevoir des notifications depuis le nuage.

- Amazon Simple Queue Service

Amazon Simple Queue Service (Amazon SQS) est un service de file d'attente de messages qui dissocie et fait évoluer les microservices, les systèmes distribués et les applications sans serveur.

- Amazon OpenSearch Service

  Amazon OpenSearch Service (OpenSearch Service) est un service géré qui facilite le déploiement, l'exploitation et le dimensionnement OpenSearch. Il s'agit d'un moteur de recherche et d'analyse open source populaire.

- Amazon SageMaker AI

  Amazon SageMaker AI peut créer des modèles d'apprentissage automatique (ML) en trouvant des modèles dans vos données IoT. Le service utilise ces modèles pour traiter de nouvelles données et générer des prévisions pour votre application.

- Amazon CloudWatch

  Amazon CloudWatch fournit une solution de surveillance fiable, évolutive et flexible pour vous aider à configurer, gérer et faire évoluer vos propres systèmes et infrastructures de surveillance.

Protocoles de communication pris en charge par AWS IoT Core

Ces rubriques fournissent plus d'informations sur les protocoles de communication utilisés par AWS IoT. Pour plus d'informations sur les protocoles utilisés par les appareils AWS IoT et les services auxquels ils connectent AWS IoT, consultezConnect à AWS IoT Core.

- MQTT (transport de télémétrie par file d'attente de messages)

  La page d'accueil du site MQTT.org où vous pouvez trouver les spécifications du protocole MQTT. Pour plus d'informations sur la prise en AWS IoT charge du MQTT, consultezMQTT.

- HTTPS (Hypertext Transfer Protocol - Secure)

  Les appareils et les applications peuvent accéder aux AWS IoT services à l'aide du protocole HTTPS.

- LoRaWAN (réseau étendu à longue portée)

  LoRaLes appareils WAN et les passerelles peuvent se connecter à AWS IoT Core l'aide AWS IoT Core du LoRa WAN.

- TLS (sécurité de la couche de transport) v1.3

La spécification du TLS v1.3 (RFC 5246). AWS IoT utilise le protocole TLS v1.3 pour établir des connexions sécurisées entre les appareils et. AWS IoT

Nouveeeeeeautés dans la nouvelleAWS IoT clé de

Nous sommes en train de mettre à jour l'interface utilisateur de laAWS IoT console pour une nouvelle expérience. Nous mettons à jour l'interface utilisateur par étapes. Ainsi, certaines pages de la console proposeront une nouvelle expérience, d'autres proposeront à la fois l'expérience d'origine et la nouvelle, et d'autres proposeront uniquement l'expérience d'origine.

Ce tableau affiche l'état des différentes zones de l'interface utilisateur de laAWS IoT console au 27 janvier 2022.

AWS IoTétat de l'interface utilisateur de la console

Page de console	de de de de de	de de de de de	Commentaires
Moniteur	Non disponible	Disponible	
Activité	Non disponible	Disponible	
À bord - Commencez	Non disponible	Disponible	Non disponible dans les régions du CN
Onboard - Modèles de provisionnement de flotte	Disponible	Disponible	
Gérer - Objets	Disponible	Disponible	
Gérer - Types	Disponible	Disponible	
Gérer : groupes d'objets	Disponible	Disponible	
Gérer - Groupes de facturation	Disponible	Disponible	

Page de console	de de de de de	de de de de de	Commentaires
Gérer - Offres d'emploi	Disponible	Disponible	
Gérer - Modèles de Job	Non disponible	Disponible	
Gérer - Tunnels	Non disponible	Disponible	
Fleet Hub - Commencez	Non disponible	Disponible	Non disponible dans tousRégions AWS
Fleet Hub - Applicati ons	Non disponible	Disponible	Non disponible dans tousRégions AWS
Greengrass - Pour commencer	Non disponible	Disponible	Non disponible dans tousRégions AWS
Greengrass - Appareils principaux	Non disponible	Disponible	Non disponible dans tousRégions AWS
Greengrass - Composants	Non disponible	Disponible	Non disponible dans tousRégions AWS
Greengrass - Déploiements	Non disponible	Disponible	Non disponible dans tousRégions AWS
Greengrass - Classique (V1)	Disponible	Disponible	
Connectivité sans fil - Intro	Non disponible	Disponible	Non disponible dans tousRégions AWS
Connectivité sans fil - Passerelles	Non disponible	Disponible	Non disponible dans tousRégions AWS
Connectivité sans fil - Appareils	Non disponible	Disponible	Non disponible dans tousRégions AWS

Page de console	de de de de de	de de de de de	Commentaires
Connectivité sans fil - Profils	Non disponible	Disponible	Non disponible dans tousRégions AWS
Connectivité sans fil - Destinations	Non disponible	Disponible	Non disponible dans tousRégions AWS
Sécurisé - Certificats	Disponible	Disponible	
Sécurisé - Politiques	Disponible	Disponible	
Sécurisé - CA	Disponible	Disponible	
Secure - Alias de rôle	Disponible	Disponible	
Sécurisé - Autorisateurs	Disponible	Disponible	
Defend - Intro	Non disponible	Disponible	
Défendez - Audit	Non disponible	Disponible	
Défendez - Détectez	Non disponible	Disponible	
Défendez - Actions d'atténuation	Non disponible	Disponible	
Defend - Paramètres	Non disponible	Disponible	
Loi - Règles	Disponible	Disponible	
Loi - Destinations	Disponible	Disponible	
Test - Device Advisor	Disponible	Disponible	Non disponible dans tousRégions AWS
Test : client de test MQTT	Disponible	Disponible	
Logiciels	Disponible	Disponible	

Page de console	de de de de de	de de de de de	Commentaires
Paramètres	Non disponible	Disponible	
Apprenez	Disponible	Pas encore disponible	

Légende

Valeurs de statut de

- Disponible

  Cette expérience d'interface utilisateur peut être utilisée.
- Non disponible

  Cette interface utilisateur ne peut pas être utilisée.
- Pas encore disponible

  La nouvelle interface utilisateur est en cours d'élaboration, mais elle n'est pas encore prête.
- En cours

  La demande de est en cours d'exécution. Certaines pages peuvent toutefois conserver l'expérience utilisateur d'origine.

Utilisation AWS IoT avec un AWS SDK

AWS des kits de développement logiciel (SDKs) sont disponibles pour de nombreux langages de programmation courants. Chaque SDK fournit une API, des exemples de code et de la documentation qui facilitent la création d’applications par les développeurs dans leur langage préféré.

Documentation SDK	Exemples de code
AWS SDK pour C++	AWS SDK pour C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK pour Go	AWS SDK pour Go exemples de code

Documentation SDK	Exemples de code
AWS SDK pour Java	AWS SDK pour Java exemples de code
AWS SDK pour JavaScript	AWS SDK pour JavaScript exemples de code
AWS SDK pour Kotlin	AWS SDK pour Kotlin exemples de code
AWS SDK pour .NET	AWS SDK pour .NET exemples de code
AWS SDK pour PHP	AWS SDK pour PHP exemples de code
Outils AWS pour PowerShell	Outils AWS pour PowerShell exemples de code
AWS SDK pour Python (Boto3)	AWS SDK pour Python (Boto3) exemples de code
AWS SDK pour Ruby	AWS SDK pour Ruby exemples de code
Kit AWS SDK pour Rust	Kit AWS SDK pour Rust exemples de code
AWS SDK pour SAP ABAP	AWS SDK pour SAP ABAP exemples de code
Kit AWS SDK pour Swift	Kit AWS SDK pour Swift exemples de code

Exemple de disponibilité

Vous n'avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code en utilisant le lien Provide feedback (Fournir un commentaire) en bas de cette page.

Commencer à utiliser les AWS IoT Core didacticiels

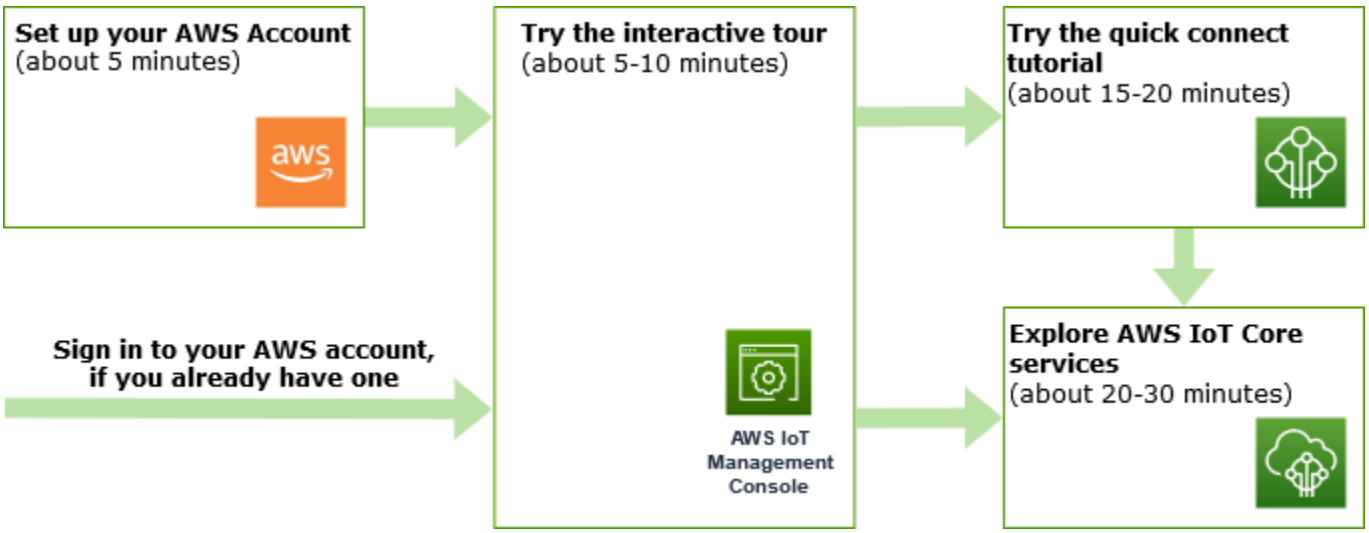
Que vous débutiez dans le domaine de l'IoT ou que vous ayez des années d'expérience, ces ressources présentent les AWS IoT concepts et les termes qui vous aideront à commencer à les utiliser AWS IoT.

- Regardez à l'intérieur AWS IoT et ses composants dedansComment AWS IoT fonctionne.
- Pour en savoir plus, AWS IoT consultez notre collection de supports de formation et de vidéos. Cette rubrique inclut également une liste de services auxquels il est AWS IoT possible de se connecter, des liens vers les réseaux sociaux et des liens vers les spécifications des protocoles de communication.
- the section called “Connectez votre premier appareil à AWS IoT Core”.
- Développez vos solutions IoT Connect à AWS IoT Core en explorant les AWS IoT tutoriels.
- Testez et validez vos appareils IoT pour une communication sécurisée et fiable en utilisant le Device Advisor.
- Gérez votre solution à l'aide de services AWS IoT Core de gestion tels que Indexation de la flotteAWS IoT Emplois, etAWS IoT Device Defender.
- Analysez les données de vos appareils à l'aide du AWS IoT services de données.

Connectez votre premier appareil à AWS IoT Core

AWS IoT Core les services connectent les appareils IoT à AWS IoT des services et à d'autres AWS services. AWS IoT Core inclut la passerelle pour appareils et le courtier de messages, qui connectent et traitent les messages entre vos appareils IoT et le cloud.

Voici comment vous pouvez commencer avec AWS IoT Core et AWS IoT.



Cette section présente une visite guidée du AWS IoT Core pour présenter ses principaux services et fournit plusieurs exemples de la manière de connecter un appareil à ceux-ci AWS IoT Core et de leur transmettre des messages. La transmission de messages entre les appareils et le cloud est essentielle à toute solution IoT et permet à vos appareils d'interagir avec d'autres AWS services.

- Configurez Compte AWS

  Avant de pouvoir utiliser AWS IoT les services, vous devez configurer un Compte AWS. Si vous avez déjà un utilisateur Compte AWS et un utilisateur IAM pour vous-même, vous pouvez les utiliser et ignorer cette étape.
- Essayez le didacticiel de connexion rapide

  Ce didacticiel est idéal si vous souhaitez démarrer rapidement AWS IoT et voir comment il fonctionne dans un scénario limité. Dans ce didacticiel, vous aurez besoin d'un appareil sur lequel vous installerez des AWS IoT logiciels. Si vous ne possédez pas d'appareil IoT, vous pouvez utiliser votre ordinateur personnel Windows, Linux ou MacOs comme appareil pour ce didacticiel. Si vous voulez essayer AWS IoT, mais que vous n'avez pas d'appareil, essayez l'option suivante.
- Essayez le tutoriel interactif

  Cette démonstration est idéale si vous souhaitez découvrir ce qu'une AWS IoT solution de base peut faire sans connecter un appareil ni télécharger de logiciel. Le didacticiel interactif présente une solution simulée basée sur AWS IoT Core des services qui illustre la manière dont ils interagissent.
- Explorez les AWS IoT Core services grâce à un didacticiel pratique

  Ce didacticiel est idéal pour les développeurs qui AWS IoT souhaitent commencer à explorer d'autres AWS IoT Core fonctionnalités, telles que le moteur de règles et les ombres. Ce didacticiel

suit un processus similaire au didacticiel de connexion rapide, mais fournit plus de détails sur chaque étape afin de permettre une transition plus fluide vers les didacticiels plus avancés.

- Afficher les messages MQTT avec le client AWS IoT MQTT

  Découvrez comment utiliser le client de test MQTT pour regarder votre premier appareil publier des messages MQTT sur AWS IoT. Le client de test MQTT est un outil utile pour surveiller et dépanner les connexions des appareils.

> **Note**
>
> Si vous souhaitez essayer plusieurs de ces didacticiels de mise en route ou répéter le même didacticiel, vous devez supprimer l'objet que vous avez créé à partir d'un didacticiel antérieur avant d'en démarrer un autre. Si vous ne supprimez pas l'objet d'un didacticiel précédent, vous devrez utiliser un nom d'objet différent pour les didacticiels suivants. En effet, le nom de l'objet doit être unique dans votre compte et Région AWS.

Pour plus d'informations AWS IoT Core, voir Qu'est-ce que c'est AWS IoT Core ?

Configurez Compte AWS

Avant de l'utiliser AWS IoT Core pour la première fois, effectuez les tâches suivantes :

Rubriques

- Inscrivez-vous pour un Compte AWS
- Création d’un utilisateur doté d’un accès administratif
- Ouvrez la AWS IoT console

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique ou un SMS et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez Utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .

2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu'utilisateur doté d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribution d'un accès à d'autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d'autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

   Pour obtenir des instructions, consultez Création d'un ensemble d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

   Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l'utilisateur AWS IAM Identity Center .

- Ouvrez la AWS IoT console

Si vous avez déjà un Compte AWS et un utilisateur pour vous-même, vous pouvez les utiliser et passer directement àthe section called "Ouvrez la AWS IoT console".

Ouvrez la AWS IoT console

La plupart des rubriques de cette section consacrées à la console commencent par la AWS IoT console. Si vous n'êtes pas encore connecté à votre Compte AWS, connectez-vous, puis ouvrez la AWS IoT console et passez à la section suivante pour continuer à démarrer AWS IoT.

Tutoriel interactif

Le didacticiel interactif présente les composants d'une solution IoT simple basée sur AWS IoT. Le didacticiel montre comment les appareils IoT interagissent avec AWS IoT Core les services. Cette rubrique fournit un aperçu du didacticiel AWS IoT Core interactif.

> **Note**
>
> Les images de la console incluent des animations qui n'apparaissent pas dans les images de ce didacticiel.

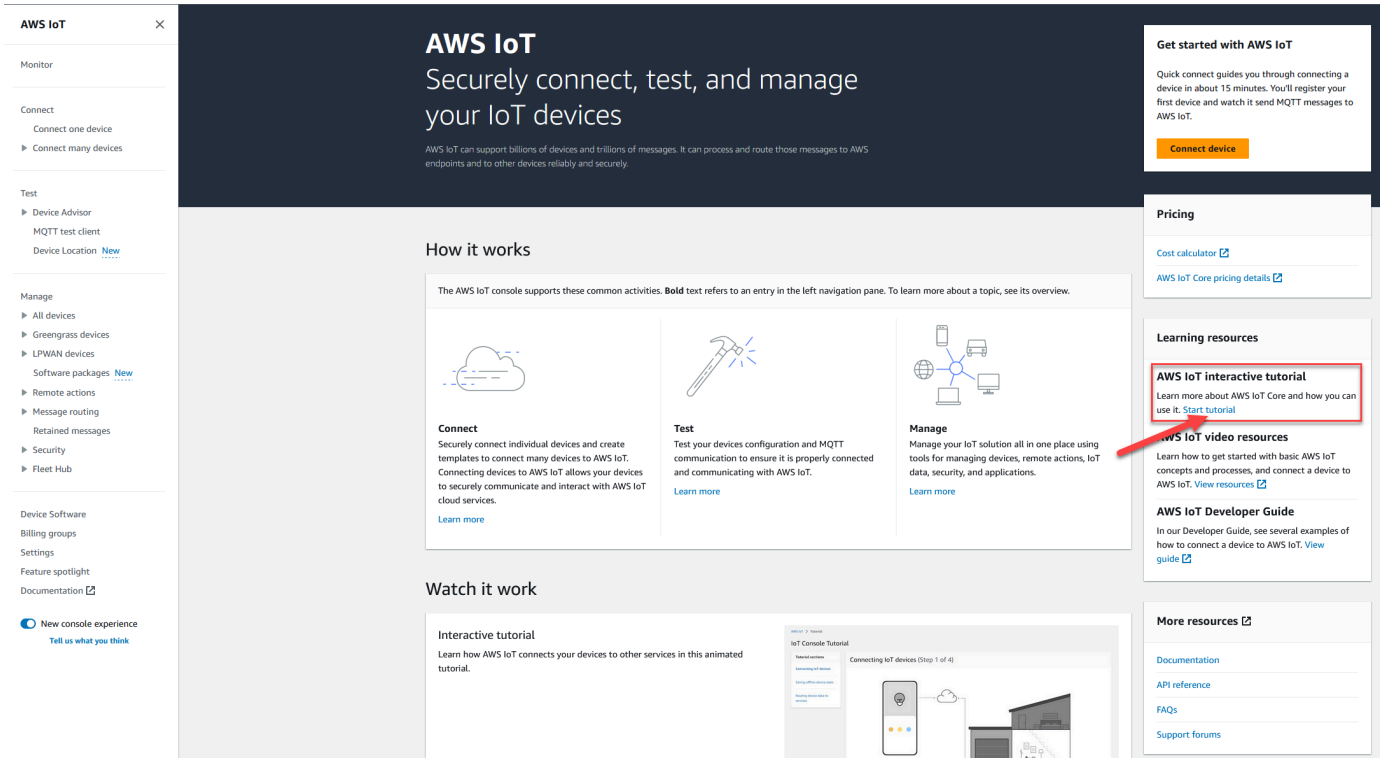
Pour lancer la démo, vous devez d'abord the section called “Configurez Compte AWS”. Le didacticiel, cependant, ne nécessite aucune AWS IoT ressource, aucun logiciel supplémentaire ou aucun codage.

Attendez-vous à consacrer environ 5 à 10 minutes à cette démonstration. En vous accordant 10 minutes, vous aurez plus de temps pour comprendre chacune des étapes.

Pour exécuter le didacticiel AWS IoT Core interactif

1. Ouvrez la page d'AWS IoT accueil dans la AWS IoT console.

   Sur la AWS IoT page d'accueil, dans le volet de la fenêtre Ressources pédagogiques, choisissez Démarrer le didacticiel.



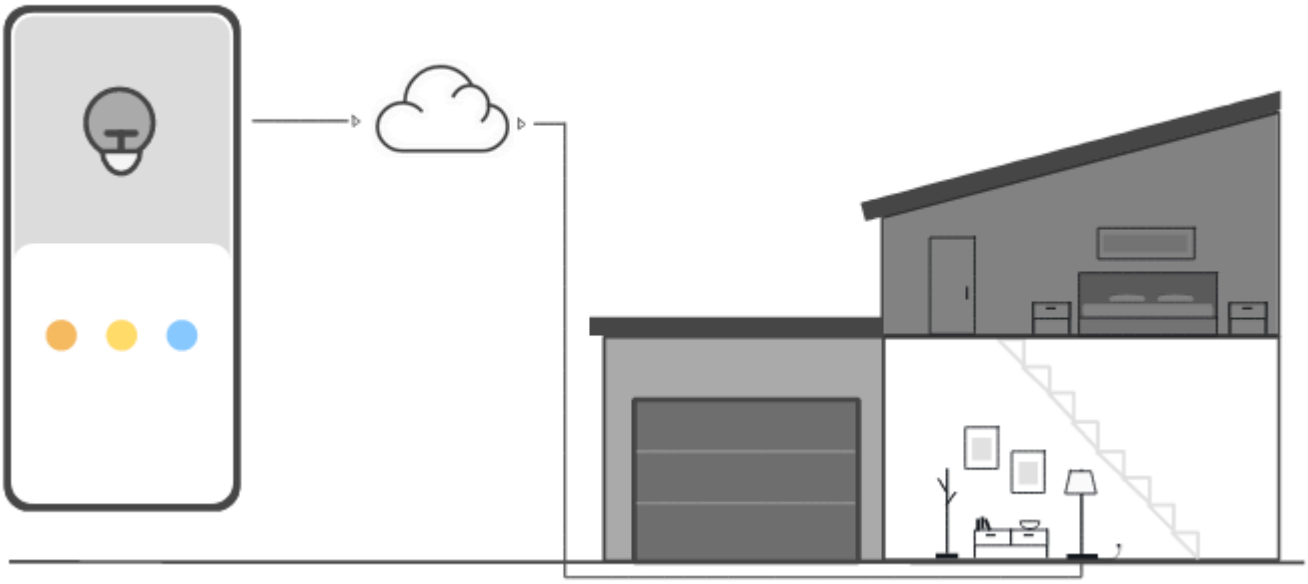
2. Sur la page AWS IoT Tutoriel de la console, passez en revue les sections du didacticiel et choisissez Démarrer la section lorsque vous êtes prêt à continuer.

Les sections suivantes décrivent comment le didacticiel de AWS IoT console présente ces AWS IoT Core fonctionnalités :

- Connexion d'appareils IoT
- Enregistrer l'état de l'appareil hors ligne
- Acheminement des données de l'appareil vers les services

Connexion d'appareils IoT

Découvrez comment les appareils IoT communiquent avec AWS IoT Core.

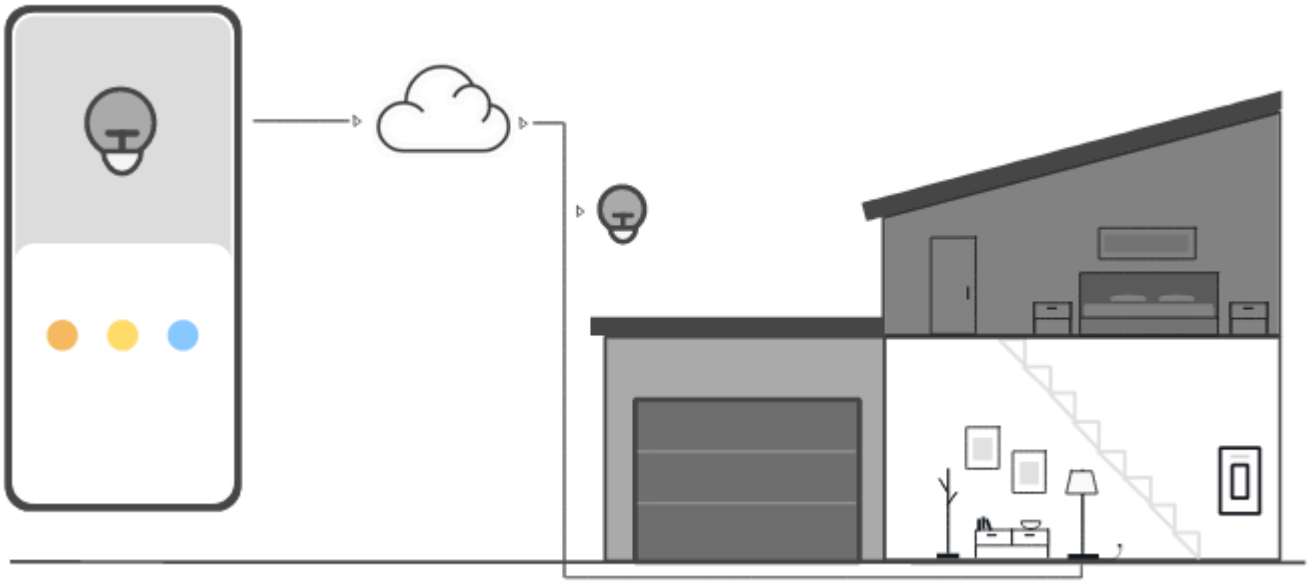


L'animation de cette étape montre comment deux appareils, le dispositif de commande sur la gauche et une lampe intelligente dans la maison sur la droite, se connectent et communiquent avec AWS IoT Core dans le cloud. L'animation montre les appareils communiquant AWS IoT Core et réagissant aux messages qu'ils reçoivent.

Pour plus d'informations sur la connexion de périphériques à AWS IoT Core, consultezConnect à AWS IoT Core.

Enregistrer l'état de l'appareil hors ligne

Découvrez comment AWS IoT Core enregistrer l'état de l'appareil lorsqu'un appareil ou une application est hors ligne.



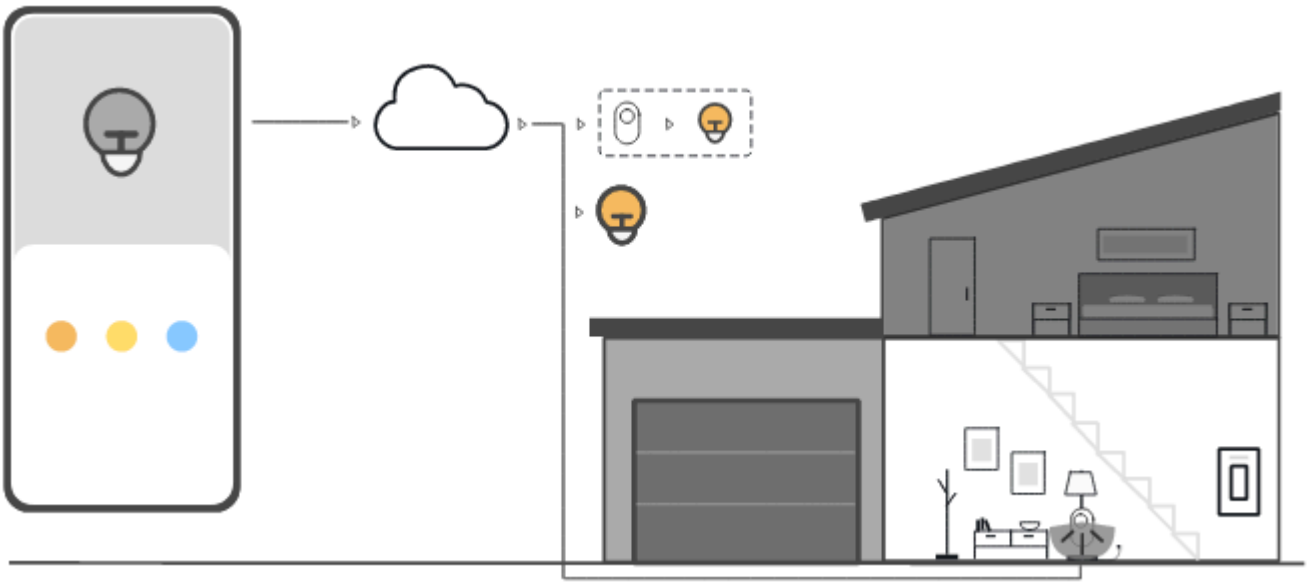
L'animation de cette étape montre comment le service Device Shadow AWS IoT Core enregistre les informations d'état de l'appareil pour le dispositif de commande et la lampe intelligente. Lorsque la lampe intelligente est hors ligne, le Device Shadow enregistre les commandes du dispositif de commande.

Lorsque la lampe intelligente se reconnecte à AWS IoT Core, elle récupère ces commandes. Lorsque le dispositif de contrôle est hors ligne, le Device Shadow enregistre les informations d'état de la lampe intelligente. Lorsque le dispositif de commande se reconnecte, il récupère l'état actuel de la lampe intelligente pour mettre à jour son affichage.

Pour plus d'informations sur Device Shadows, consultez AWS IoT Service Device Shadow.

Acheminement des données de l'appareil vers les services

Découvrez comment AWS IoT Core envoyer l'état de l'appareil à d'autres AWS services.



L'animation de cette étape montre comment AWS IoT Core envoyer les données des appareils à d'autres AWS services à l'aide de AWS IoT règles. AWS IoT les règles s'abonnent à des messages spécifiques provenant des appareils, interprètent les données contenues dans ces messages et acheminent les données interprétées vers d'autres services. Dans cet exemple, une AWS IoT règle interprète les données d'un détecteur de mouvement et envoie des commandes à un Device Shadow, qui les envoie ensuite à l'ampoule intelligente. Comme dans l'exemple précédent, le Device Shadow stocke les informations relatives à l'état de l'appareil de commande.

Pour plus d'informations sur AWS IoT les règles, consultezRègles pour AWS IoT.

Essayez le didacticiel de connexion AWS IoT Core rapide

Dans ce didacticiel, vous allez créer votre premier objet, y connecter un appareil et le regarder envoyer des messages MQTT.

Attendez à consacrer 15 à 20 minutes à ce didacticiel.

Ce didacticiel est idéal pour les personnes qui souhaitent se lancer rapidement AWS IoT pour voir comment il fonctionne dans un scénario limité. Si vous recherchez un exemple qui vous permettra de démarrer et d'explorer davantage de fonctionnalités et de services, essayez Explorez AWS IoT Core dans des didacticiels pratiques.

Dans ce didacticiel, vous allez télécharger et exécuter un logiciel sur un appareil qui se connecte à une ressource d'objets dans AWS IoT Core le cadre d'une très petite solution IoT. L'appareil peut être

un appareil IoT, tel qu'un Raspberry Pi, ou il peut également s'agir d'un ordinateur exécutant Linux, OS et OSX, ou Windows. Si vous souhaitez connecter un périphérique WAN (LoRaWAN) à longue portée AWS IoT, reportez-vous au didacticiel >Connecter des appareils et des passerelles au AWS IoT Core WAN. LoRa

Si votre appareil est compatible avec un navigateur capable d'exécuter la AWS IoT console, nous vous recommandons de suivre ce didacticiel sur cet appareil.

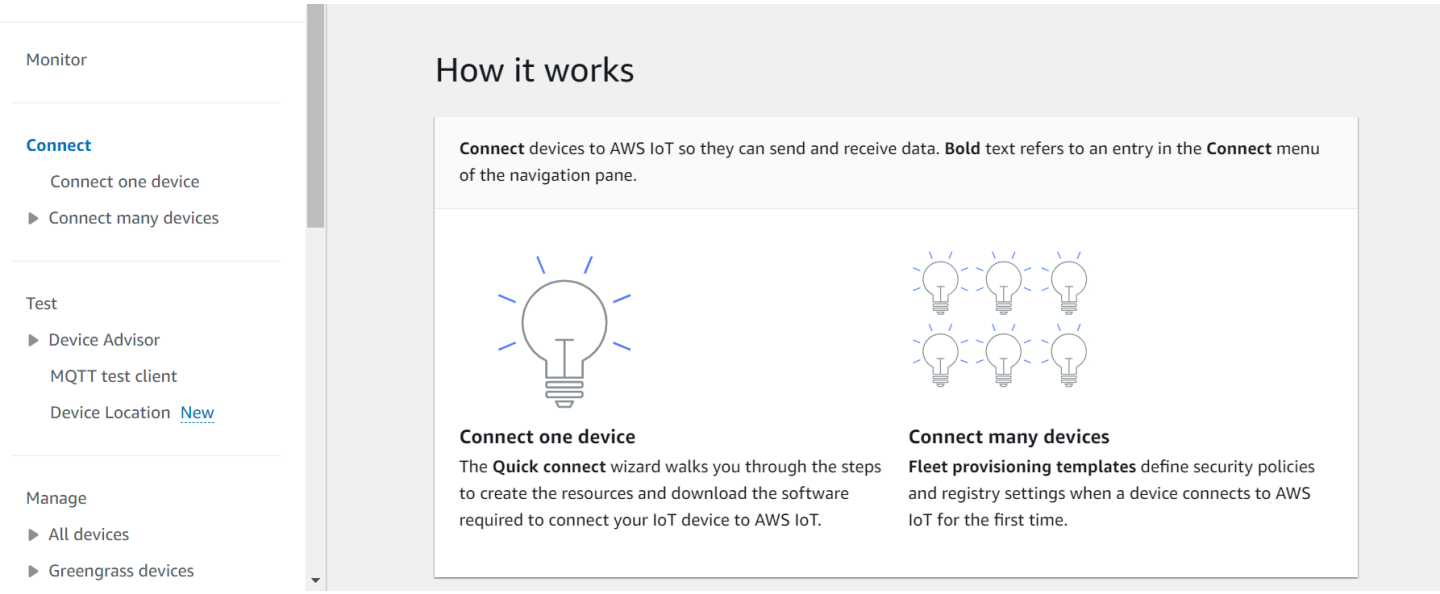
> **Note**
>
> Si votre appareil n'est pas doté d'un navigateur compatible, suivez ce didacticiel sur ordinateur. Lorsque la procédure vous demande de télécharger le fichier, téléchargez-le sur votre ordinateur, puis transférez le fichier téléchargé sur votre appareil à l'aide de Secure Copy (SCP) ou d'un processus similaire.

Le didacticiel nécessite que votre appareil IoT communique avec le port 8443 sur le point de terminaison Compte AWS de données de votre appareil. Pour vérifier s'il peut accéder à ce port, essayez les procédures décrites dansTestez la connectivité avec le point de terminaison de données de votre appareil.

Étape 1. Commencez avec le didacticiel

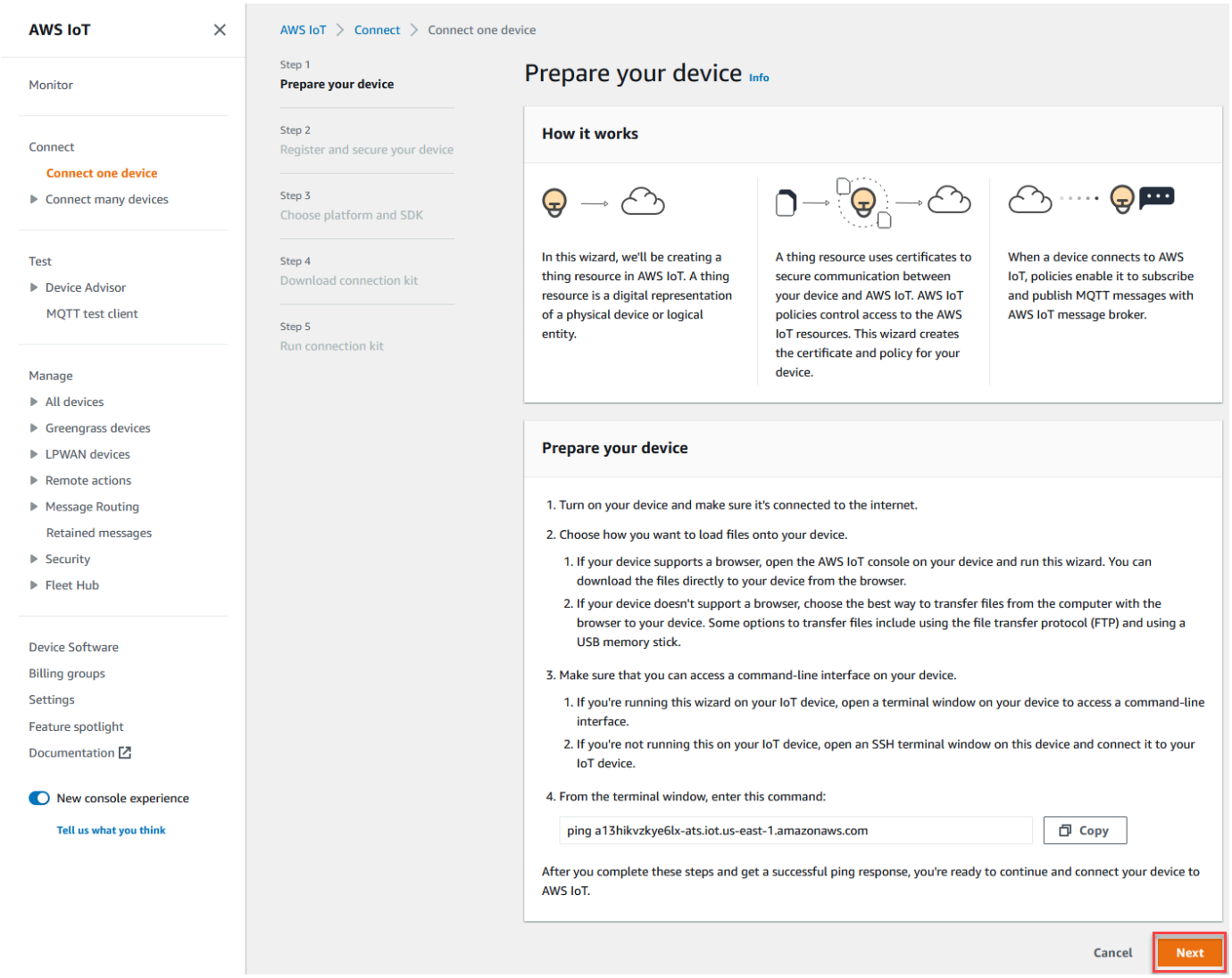
Si possible, effectuez cette procédure sur votre appareil ; sinon, soyez prêt à transférer un fichier sur votre appareil plus tard dans cette procédure.

Pour démarrer le didacticiel, connectez-vous à la AWS IoT console. Sur la page d'accueil de la AWS IoT console, sur la gauche, choisissez Connect, puis Connect one device.



Étape 2. Cela crée un objet

1. Dans la section Préparer votre appareil, suivez les instructions affichées à l'écran pour préparer votre appareil à la connexion à AWS IoT.

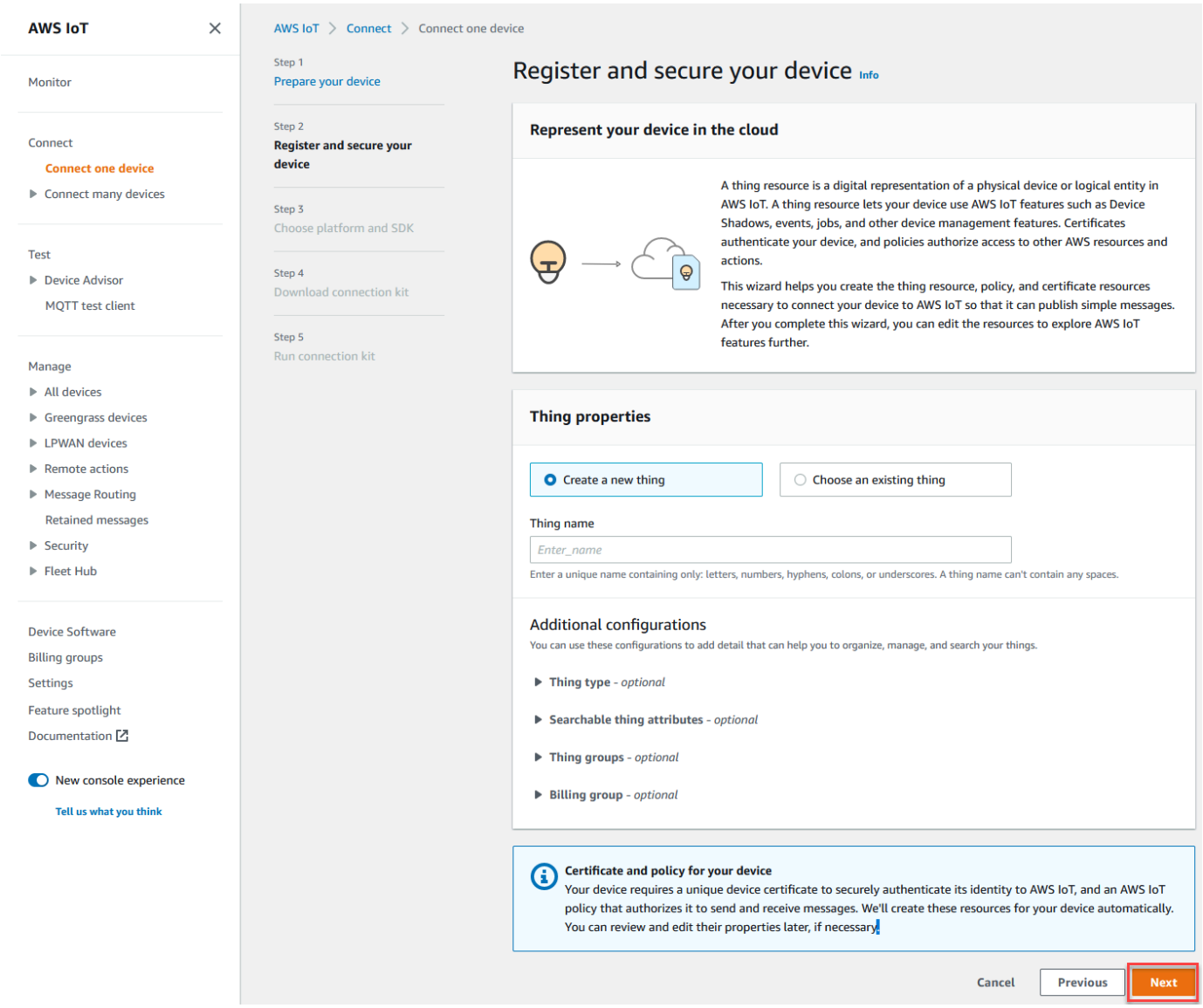


2. Dans la section Enregistrer et sécuriser votre appareil, choisissez Créer un nouvel objet ou Choisir un objet existant. Dans le champ Nom de l'objet, entrez le nom de votre objet. Le nom de l'objet utilisé dans cet exemple est **`TutorialTestThing`**

> **Important**
>
> Vérifiez le nom de votre objet avant de continuer.
> Vous ne pouvez plus modifier le nom d'un objet. Si vous souhaitez modifier le nom d'un objet, vous devez créer un objet portant le nom d'objet correct, puis supprimer celui dont le nom est incorrect.

Dans la section Configurations supplémentaires, personnalisez davantage votre ressource d'objets à l'aide des configurations facultatives répertoriées.

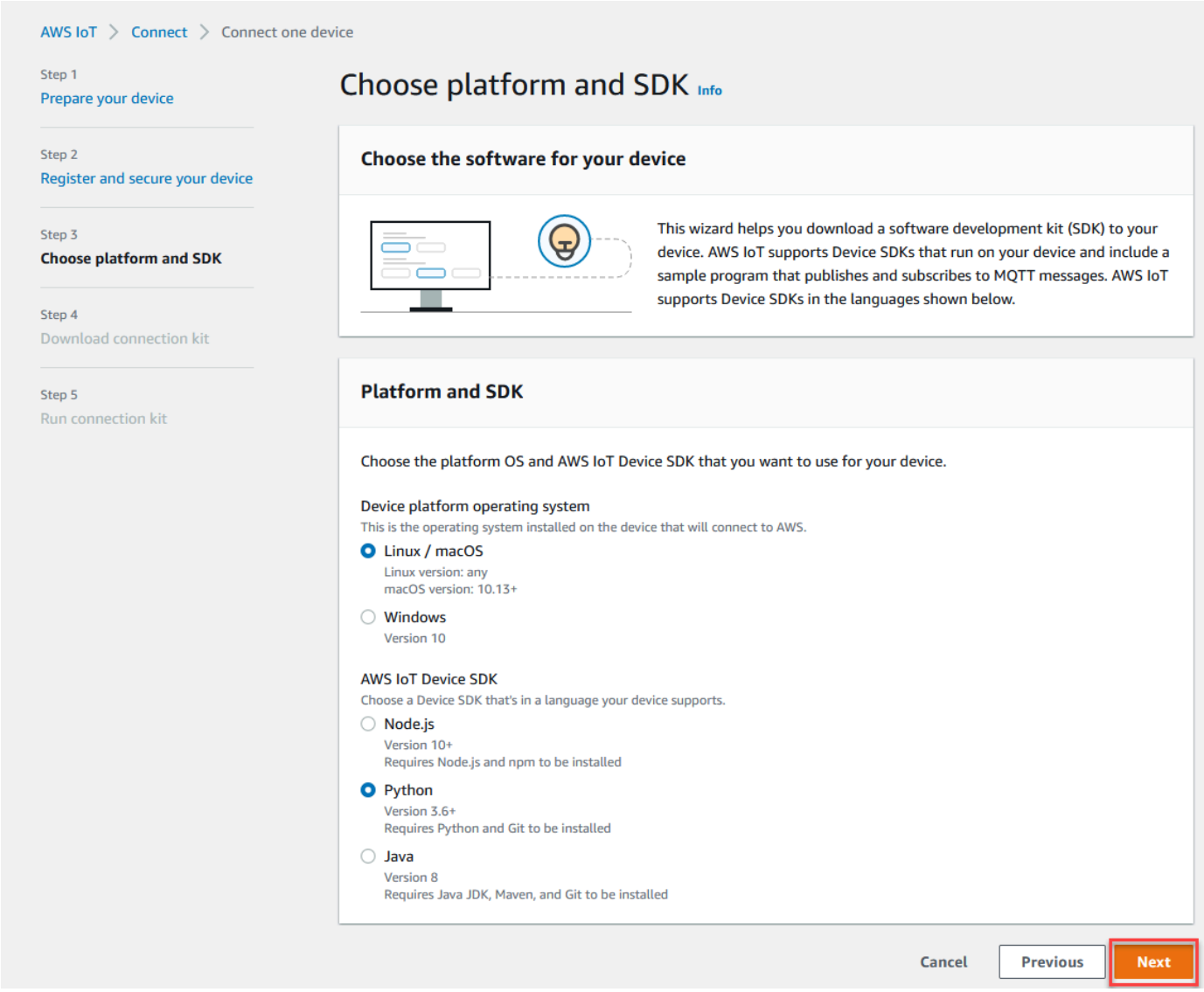
Après avoir donné un nom à votre objet et sélectionné des configurations supplémentaires, choisissez Suivant.



3. Dans la section Choisir la plate-forme et le SDK, choisissez la plate-forme et la langue du SDK du AWS IoT périphérique que vous souhaitez utiliser. Cet exemple utilise la Linux/OSX plateforme et le SDK Python. Assurez-vous que python3 et pip3 sont installés sur votre appareil cible avant de passer à l'étape suivante.

> **Note**
>
> Assurez-vous de consulter la liste des logiciels requis par le SDK que vous avez choisi au bas de la page de console.
> Le logiciel requis doit être installé sur votre ordinateur cible avant de passer à l'étape suivante.

Après avoir choisi la langue du SDK de la plate-forme et de l'appareil, choisissez Suivant.

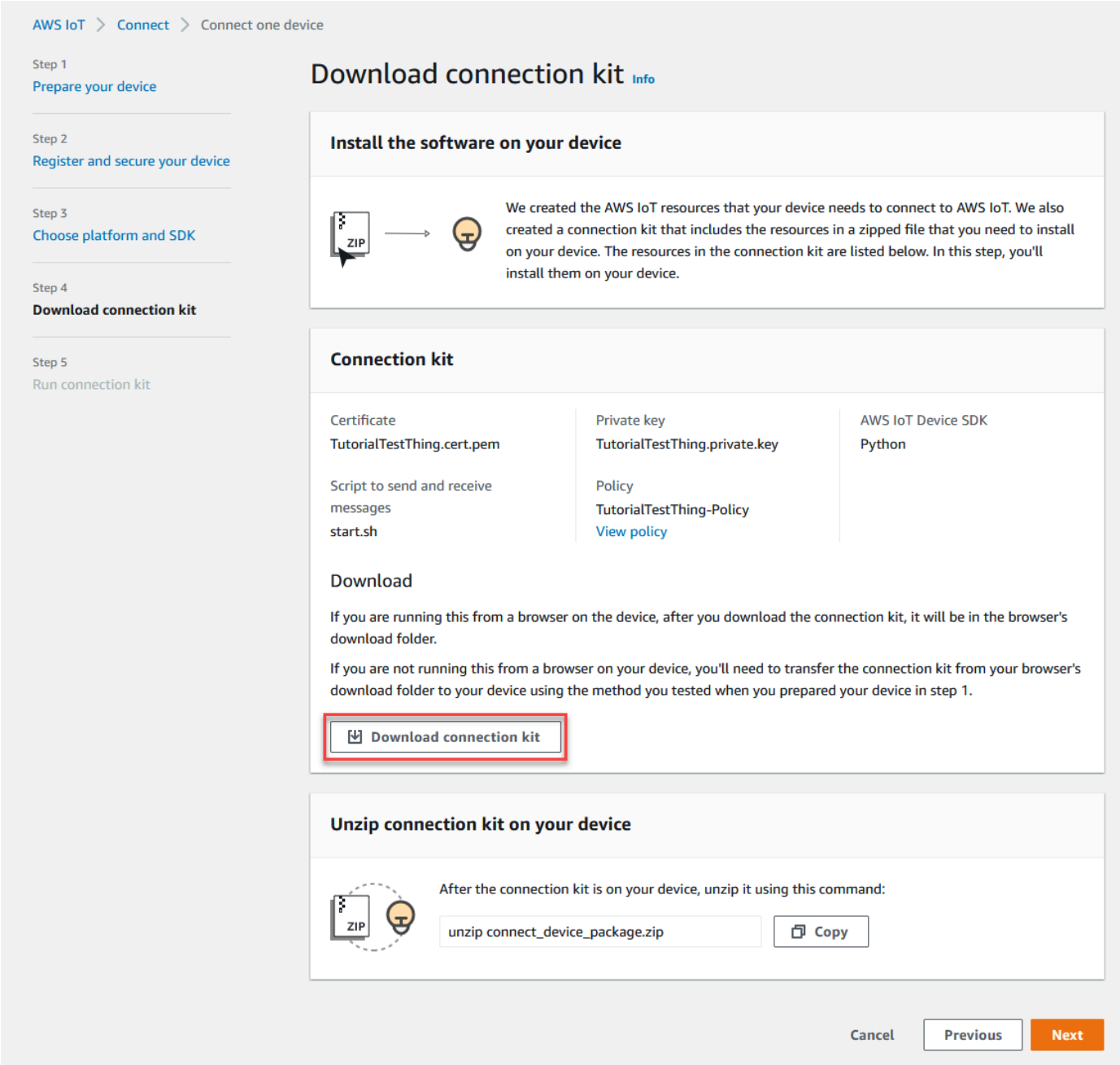


Étape 3. Téléchargez des fichiers sur votre appareil

Cette page apparaît après AWS IoT la création du kit de connexion, qui inclut les fichiers et ressources suivants dont votre appareil a besoin :

- Les fichiers de certificat utilisés pour authentifier l'appareil
- Une ressource politique pour autoriser votre objet à interagir avec AWS IoT
- Le script permettant de télécharger le SDK du AWS périphérique et d'exécuter l'exemple de programme sur votre appareil

1. Lorsque vous êtes prêt à continuer, cliquez sur le bouton Télécharger le kit de connexion pour télécharger le kit de connexion pour la plate-forme que vous avez choisie précédemment.



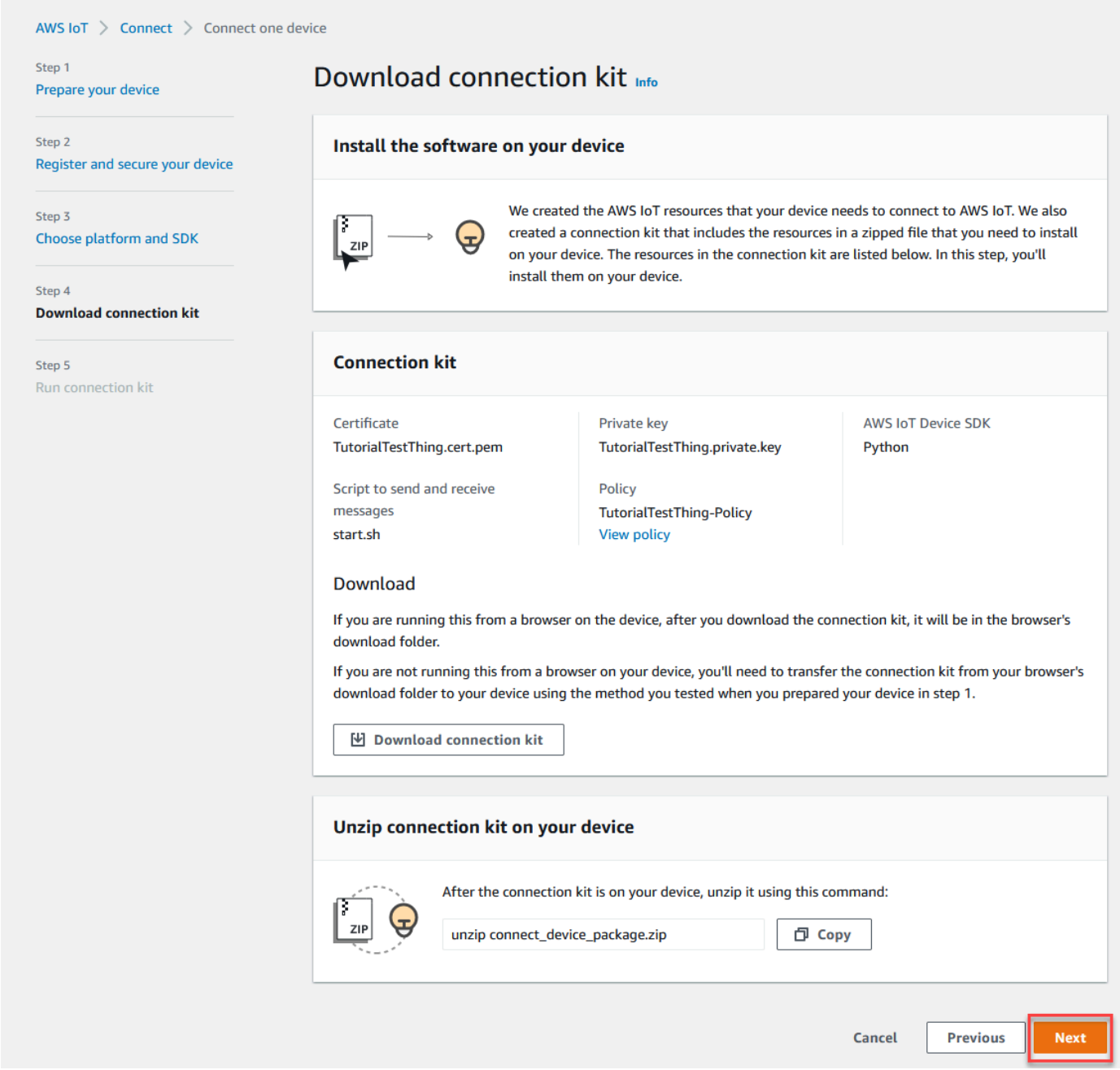
2. Si vous exécutez cette procédure sur votre appareil, enregistrez le fichier du kit de connexion dans un répertoire à partir duquel vous pouvez exécuter des commandes en ligne.

   Si vous n'exécutez pas cette procédure sur votre appareil, enregistrez le fichier du kit de connexion dans un répertoire local, puis transférez-le sur votre appareil.

3. Dans la section Décompressez le kit de connexion sur votre appareil, entrez unzip connect_device_package.zip dans le répertoire où se trouvent les fichiers du kit de connexion.

Si vous utilisez une fenêtre de PowerShell commande Windows et que la unzip commande ne fonctionne pas, unzip remplacez-la expand-archive par et réessayez la ligne de commande.

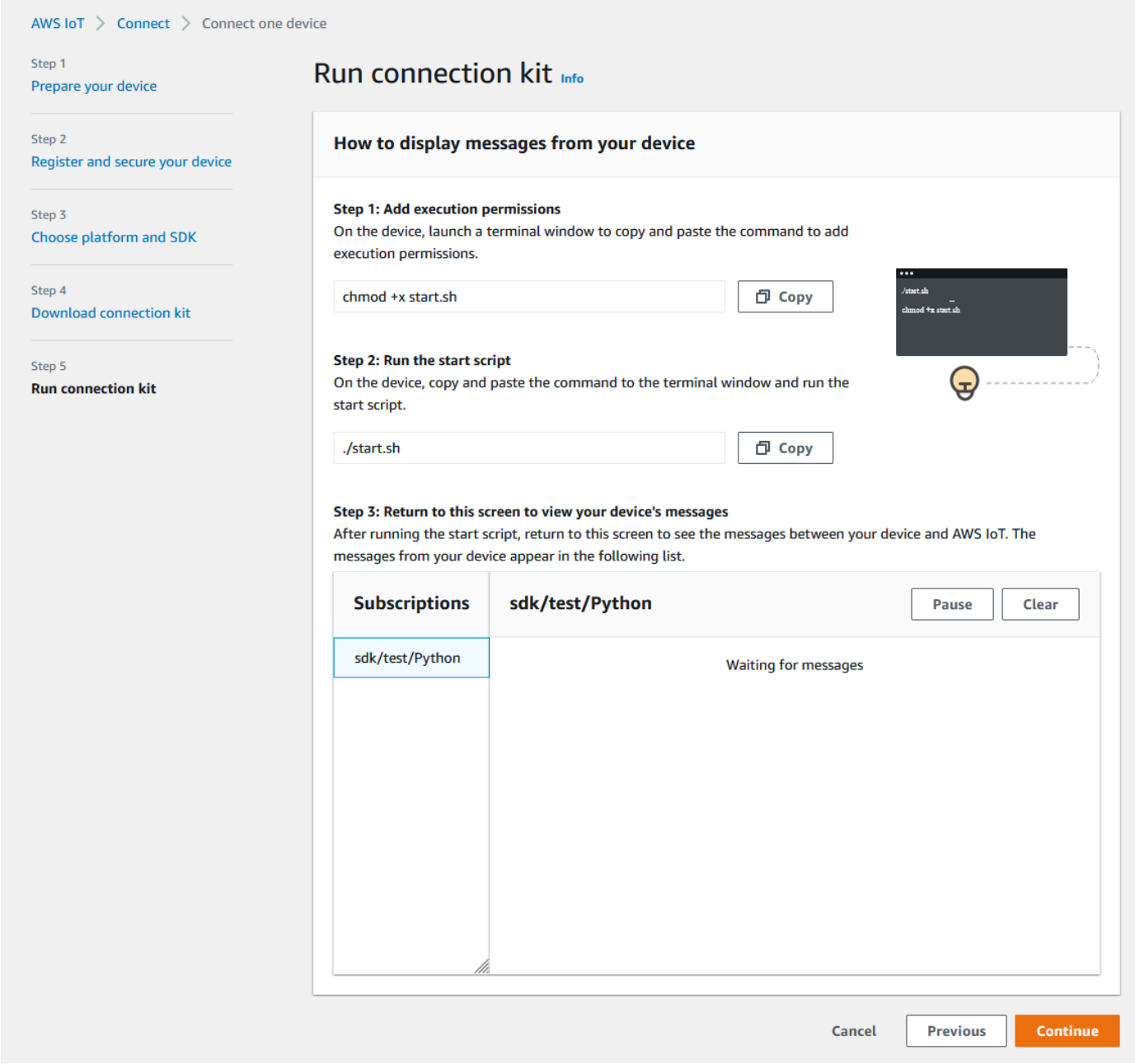
4. Après avoir installé le fichier du kit de connexion sur l'appareil, poursuivez le didacticiel en choisissant Suivant.



Étape 4 : Exécutez l'échantillon

Vous devez effectuer cette procédure dans un terminal ou une fenêtre de commande de votre appareil tout en suivant les instructions affichées dans la console. Les commandes affichées dans la console correspondent au système d'exploitation que vous avez choisi dans the section called “Étape 2. Cela crée un objet”. Celles présentées ici concernent les systèmes Linux/OSX d'exploitation.

1. Dans un terminal ou une fenêtre de commande de votre appareil, dans le répertoire contenant le fichier du kit de connexion, effectuez les étapes indiquées dans la AWS IoT console.



2. Après avoir saisi la commande de l'étape 2 dans la console, vous devriez voir un résultat similaire à ce qui suit dans le terminal ou dans la fenêtre de commande de l'appareil. Ce résultat provient des messages que le programme envoie puis reçoit en retour AWS IoT Core.

```
Running pub/sub sample application...
Connecting to a13hikvzkye6lx-ats.iot.us-east-1.amazonaws.com with client ID 'basicPubSub'...
Connected!
Subscribing to topic 'sdk/test/Python'...
Subscribed with QoS.AT_LEAST_ONCE
Sending messages until program killed
Publishing message to topic 'sdk/test/Python': Hello World! [1]
Received message from topic 'sdk/test/Python': b'"Hello World! [1]"'
Publishing message to topic 'sdk/test/Python': Hello World! [2]
Received message from topic 'sdk/test/Python': b'"Hello World! [2]"'
Publishing message to topic 'sdk/test/Python': Hello World! [3]
Received message from topic 'sdk/test/Python': b'"Hello World! [3]"'
```

   Pendant l'exécution de l'exemple de programme, le message de test `Hello World!` apparaît également. Le message de test apparaît dans le terminal ou dans la fenêtre de commande de votre appareil.

   > **Note**
   >
   > Pour plus d'informations sur l'abonnement aux rubriques et leur publication, consultez l'exemple de code du SDK que vous avez choisi.

3. Pour exécuter à nouveau l'exemple de programme, vous pouvez répéter les commandes de l'étape 2 dans la console de cette procédure.
4. (Facultatif) Si vous souhaitez voir les messages de votre client IoT dans la AWS IoT console, ouvrez le client de test MQTT sur la page Test de la AWS IoT console. Si vous avez choisi le SDK Python, dans le client de test MQTT, dans le filtre d'objet, entrez l'objet, par exemple **`sdk/test/python`** pour vous abonner aux messages depuis votre appareil. Les filtres thématiques distinguent les majuscules et minuscules et dépendent du langage de programmation du SDK que vous avez choisi à l'étape 1. Pour plus d'informations sur l'abonnement aux rubriques et leur publication, consultez l'exemple de code du SDK que vous avez choisi.
5. Après vous être abonné à la rubrique de test, exécutez-le ./start.sh sur votre appareil. Pour de plus amples informations, veuillez consulter the section called "Afficher les messages MQTT avec le client AWS IoT MQTT".

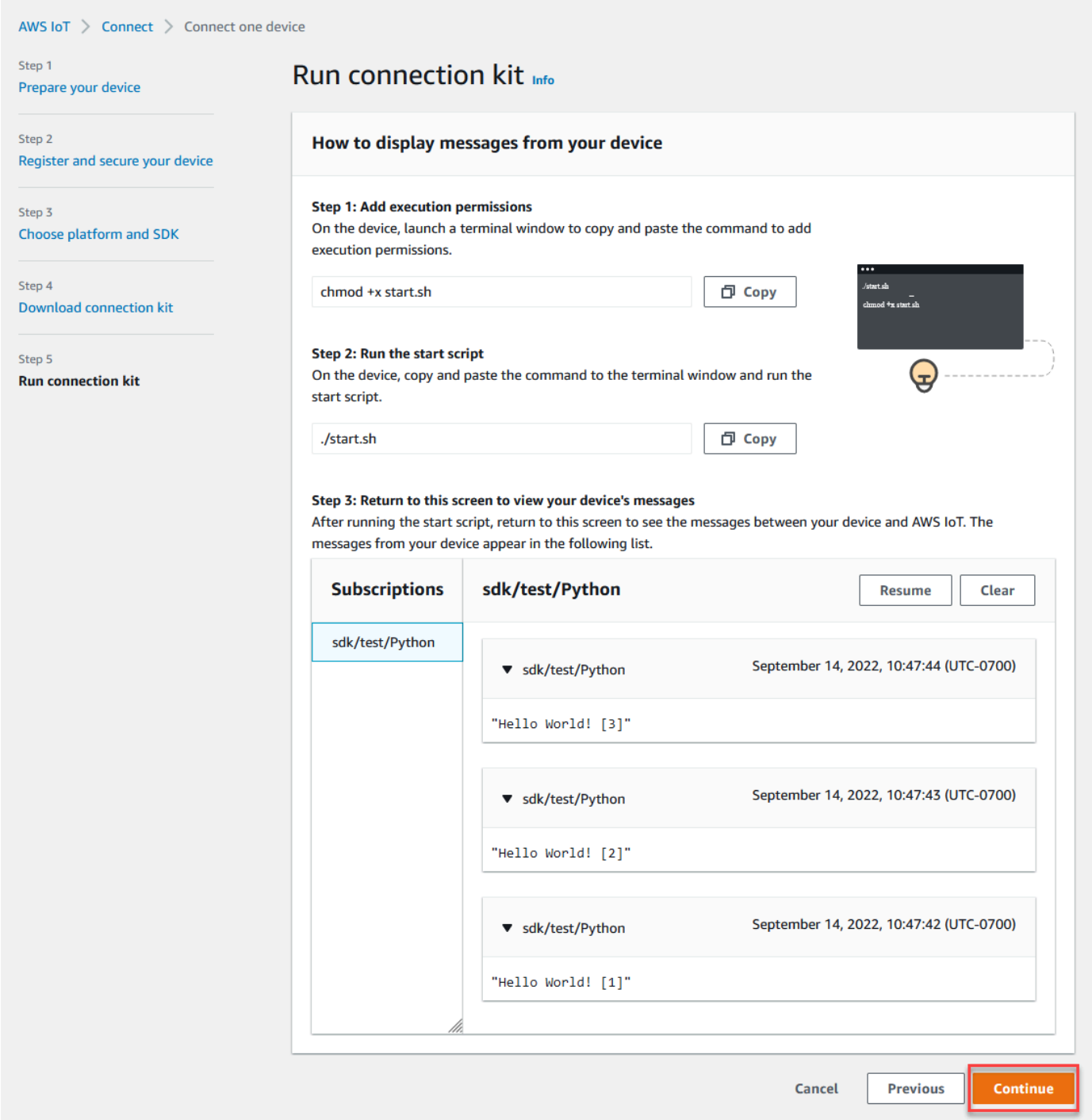
   Après l'exécution de ./start.sh, des messages similaires aux suivants apparaissent dans le client MQTT :

```
{
```

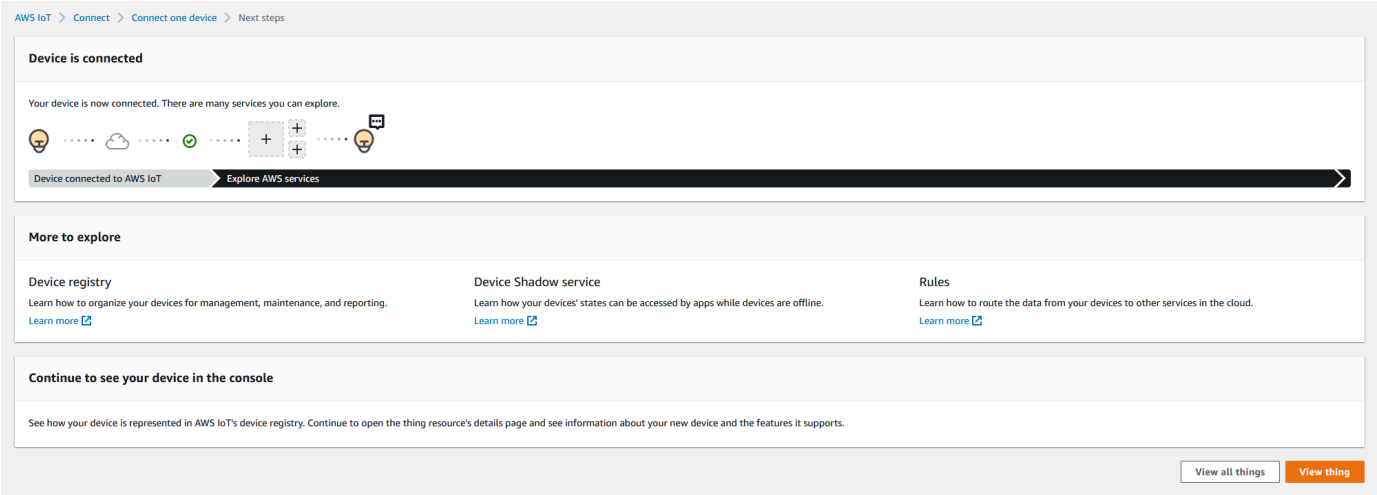
```
  "message": "Hello World!" [1]
}
```

Le nombre `sequence` est incrémenté dans `[]` d'une unité chaque fois qu'un nouveau message `Hello World!` est reçu et s'arrête lorsque vous terminez le programme.

6. Pour terminer le didacticiel et voir un résumé, dans la AWS IoT console, choisissez Continuer.



7. Un résumé de votre didacticiel de connexion AWS IoT rapide s'affiche désormais.



Étape 5. Explorez davantage

Voici quelques idées à AWS IoT approfondir une fois que vous aurez terminé le démarrage rapide.

- Afficher les messages MQTT dans le client MQTT

  Depuis la AWS IoT console, vous pouvez ouvrir le client MQTT sur la page Test de la console AWS IoT . Dans le client de test MQTT, abonnez-vous au programme **#**, puis exécutez le programme ./start.sh sur votre appareil comme décrit à l'étape précédente. Pour de plus amples informations, veuillez consulter the section called “Afficher les messages MQTT avec le client AWS IoT MQTT”.
- Exécutez des tests sur vos appareils avec Device Advisor

  Utilisez Device Advisor pour vérifier si vos appareils peuvent se connecter et interagir de manière sûre et fiable avec AWS IoT.
- the section called “Tutoriel interactif”

  Pour démarrer le didacticiel interactif, sur la page Apprendre de la AWS IoT console, dans la vignette Voir comment AWS IoT fonctionne, sélectionnez Démarrer le didacticiel.
- Préparez-vous à découvrir d'autres didacticiels

  Ce démarrage rapide ne vous donne qu'un échantillon de AWS IoT. Si vous souhaitez en savoir AWS IoT plus et en savoir plus sur les fonctionnalités qui en font une puissante plateforme de solutions IoT, commencez à préparer votre plateforme de développement enExplorez AWS IoT Core dans des didacticiels pratiques.

Testez la connectivité avec le point de terminaison de données de votre appareil

Cette rubrique explique comment tester la connexion d'un appareil avec le point de terminaison de données de votre compte, le point de terminaison auquel vos appareils IoT se connectent AWS IoT.

Effectuez ces procédures sur le périphérique que vous souhaitez tester ou en utilisant une session de terminal SSH connectée à l'appareil que vous souhaitez tester.

Pour tester la connectivité d'un appareil avec le point de terminaison de données de votre appareil.

- Trouvez le point de terminaison des données de votre appareil
- Testez rapidement la connexion
- Téléchargez l'application pour tester la connexion aux données, au point de terminaison et au port de votre appareil
- Testez la connexion avec le point de terminaison et le port de données de votre appareil

Trouvez le point de terminaison des données de votre appareil

Cette procédure explique comment trouver le point de terminaison des données de votre appareil dans la AWS IoT console pour tester la connexion à votre appareil IoT.

Pour trouvez le point de terminaison des données de votre appareil

1. Dans la AWS IoT console, dans la section Connect, accédez à Configurations de domaine.
2. Sur la page Configurations de domaine, accédez au conteneur de configurations de domaine et copiez le nom de domaine. La valeur de votre point de terminaison est unique à votre Compte AWS et est similaire à cet exemple :`a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`.
3. Enregistrez le point de terminaison des données de votre appareil à utiliser dans les procédures suivantes.

Testez rapidement la connexion

Cette procédure teste la connectivité générale avec le point de terminaison de données de votre appareil, mais elle ne teste pas le port spécifique que vos appareils utiliseront. Ce test utilise un

programme courant et est généralement suffisant pour savoir si vos appareils peuvent se connecter à AWS IoT.

Si vous souhaitez tester la connectivité avec le port spécifique que vos appareils utiliseront, ignorez cette procédure et passez à Téléchargez l'application pour tester la connexion aux données, au point de terminaison et au port de votre appareil.

Pour tester rapidement le point de terminaison des données de l'appareil

1. Dans une fenêtre de terminal ou de ligne de commande de votre appareil, remplacez l'exemple de point de terminaison de données de l'appareil (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) par le point de terminaison de données de l'appareil pour votre compte, puis entrez cette commande.

   Linux

   ```
   ping -c 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

   Windows

   ```
   ping -n 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

2. S'il `ping` affiche une sortie similaire à la suivante, il s'est correctement connecté au point de terminaison de données de votre appareil. Bien qu'il n'ait pas communiqué AWS IoT directement avec lui, il a trouvé le serveur et il est probable qu'il AWS IoT soit disponible via ce point de terminaison.

   ```
   PING a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (xx.xx.xxx.xxx) 56(84) bytes of
    data.
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=1 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=2 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=3 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=4 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=5 ttl=231 time=127 ms
   ```

Si vous êtes satisfait de ce résultat, vous pouvez arrêter les tests ici.

Si vous souhaitez tester la connectivité avec le port spécifique utilisé par AWS IoT, passez à Téléchargez l'application pour tester la connexion aux données, au point de terminaison et au port de votre appareil.

3. Si `ping` n'a pas renvoyé de résultat réussi, vérifiez la valeur du point de terminaison pour vous assurer que vous disposez du bon point de terminaison et vérifiez la connexion de l'appareil avec Internet.

Téléchargez l'application pour tester la connexion aux données, au point de terminaison et au port de votre appareil

Un test de connectivité plus approfondi peut être effectué en utilisant `nmap`. Cette procédure permet de vérifier si `nmap` est installé sur votre appareil.

Pour vérifier **`nmap`** sur l'appareil

1. Dans un terminal ou une fenêtre de ligne de commande de l'appareil que vous souhaitez tester, entrez cette commande pour voir si `nmap` est installée.

```
nmap --version
```

2. Si vous obtenez une sortie similaire à ce qui suit, `nmap` est installée et vous pouvez continuer vers the section called “Testez la connexion avec le point de terminaison et le port de données de votre appareil”.

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

3. Si vous ne voyez pas de réponse similaire à celle indiquée à l'étape précédente, vous devez procéder à l'installation de `nmap` sur l'appareil. Choisissez la procédure pour le système d'exploitation de votre appareil.

Linux

Cette procédure nécessite que vous ayez l'autorisation d'installer le logiciel sur l'ordinateur.

Pour installer nmap sur votre ordinateur Linux

1. Dans un terminal ou une fenêtre de ligne de commande de votre appareil, entrez la commande correspondant à la version de Linux qu'il exécute.

   a. Pour Debian/Ubuntu :

   ```
   sudo apt install nmap
   ```

   b. Pour CentOS ou RHEL :

   ```
   sudo yum install nmap
   ```

2. Testez l'installation avec cette commande :

   ```
   nmap --version
   ```

3. Si vous obtenez une sortie similaire à ce qui suit, nmap est installée et vous pouvez continuer vers the section called “Testez la connexion avec le point de terminaison et le port de données de votre appareil”.

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

macOS

Cette procédure nécessite que vous ayez l'autorisation d'installer le logiciel sur l'ordinateur.

Pour installer nmap sur votre ordinateur MacOs

1. Dans un navigateur, ouvrez https://nmap.org/download#macosx et téléchargez la dernière version stable du programme.

Lorsque vous y êtes invité, sélectionnez Ouvrir avec DiskImageInstaller.

2. Dans la fenêtre d'installation, déplacez le package vers le dossier Applications.
3. Dans le Recherche, recherchez le `nmap-xxxx-mpkg` package dans le dossier Applications. Ctrl-clickcliquez sur le package et sélectionnez Ouvrir pour ouvrir le package.
4. Consultez la boîte de dialogue de sécurité. Si vous êtes prêt à procéder à l'installationnmap, choisissez Ouvrir pour effectuer l'installationnmap.
5. Dans Terminal, testez l'installation avec cette commande.

```
nmap --version
```

6. Si vous obtenez une sortie similaire à ce qui suit, `nmap` est installée et vous pouvez continuer vers the section called “Testez la connexion avec le point de terminaison et le port de données de votre appareil”.

```
Nmap version 7.92 ( https://nmap.org )
Platform: x86_64-apple-darwin17.7.0
Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 libz-1.2.11
 nmap-libpcre-7.6 nmap-libpcap-1.9.1 nmap-libdnet-1.12 ipv6 Compiled without:
Available nsock engines: kqueue poll select
```

Windows

Cette procédure nécessite que vous ayez l'autorisation d'installer le logiciel sur l'ordinateur.

Pour installer nmap sur votre ordinateur Windows

1. Dans un navigateur, ouvrez https://nmap.org/download#windows et téléchargez la dernière version stable du programme d'installation.

   Si vous y êtes invité, choisissez Enregistrer le fichier. Une fois le fichier téléchargé, ouvrez-le depuis le dossier des téléchargements.
2. Une fois le téléchargement du fichier d'installation terminé, ouvrez le fichier téléchargé nmap-xxxx-setup.exe pour installer l'application.
3. Acceptez les paramètres par défaut lors de l'installation du programme.

   Vous n'avez pas besoin de l'application Npcap pour ce test. Vous pouvez désélectionner cette option si vous ne souhaitez pas l'installer.

4. Dans Command, testez l'installation avec cette commande.

```
nmap --version
```

5. Si vous obtenez une sortie similaire à ce qui suit, nmap est installée et vous pouvez continuer vers the section called “Testez la connexion avec le point de terminaison et le port de données de votre appareil”.

```
Nmap version 7.92 ( https://nmap.org )
Platform: i686-pc-windows-windows
Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 nmap-
libz-1.2.11 nmap-libpcre-7.6 Npcap-1.50 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: iocp poll select
```

Testez la connexion avec le point de terminaison et le port de données de votre appareil

Cette procédure teste la connexion de votre appareil IoT au point de terminaison de données de votre appareil à l'aide du port que vous avez sélectionné.

Pour tester le point de terminaison et le port de données de votre appareil

1. Dans une fenêtre de terminal ou de ligne de commande de votre appareil, remplacez l'exemple de point de terminaison de données de l'appareil (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) par le point de terminaison de données de l'appareil pour votre compte, puis entrez cette commande.

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

2. Si nmap affiche une sortie similaire ceci, nmap a réussi à se connecter au point de terminaison de données de votre appareil sur le port sélectionné.

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-18 16:23 Pacific Standard Time
Nmap scan report for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
 (xx.xxx.147.160)
Host is up (0.036s latency).
Other addresses for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (not scanned):
 xx.xxx.134.144 xx.xxx.55.139 xx.xxx.110.235 xx.xxx.174.233 xx.xxx.74.65
 xx.xxx.122.179 xx.xxx.127.126
```

```
rDNS record for xx.xxx.147.160: ec2-EXAMPLE-160.eu-west-1.compute.amazonaws.com

PORT     STATE SERVICE
8443/tcp open  https-alt
MAC Address: 00:11:22:33:44:55 (Cimsys)

Nmap done: 1 IP address (1 host up) scanned in 0.91 seconds
```

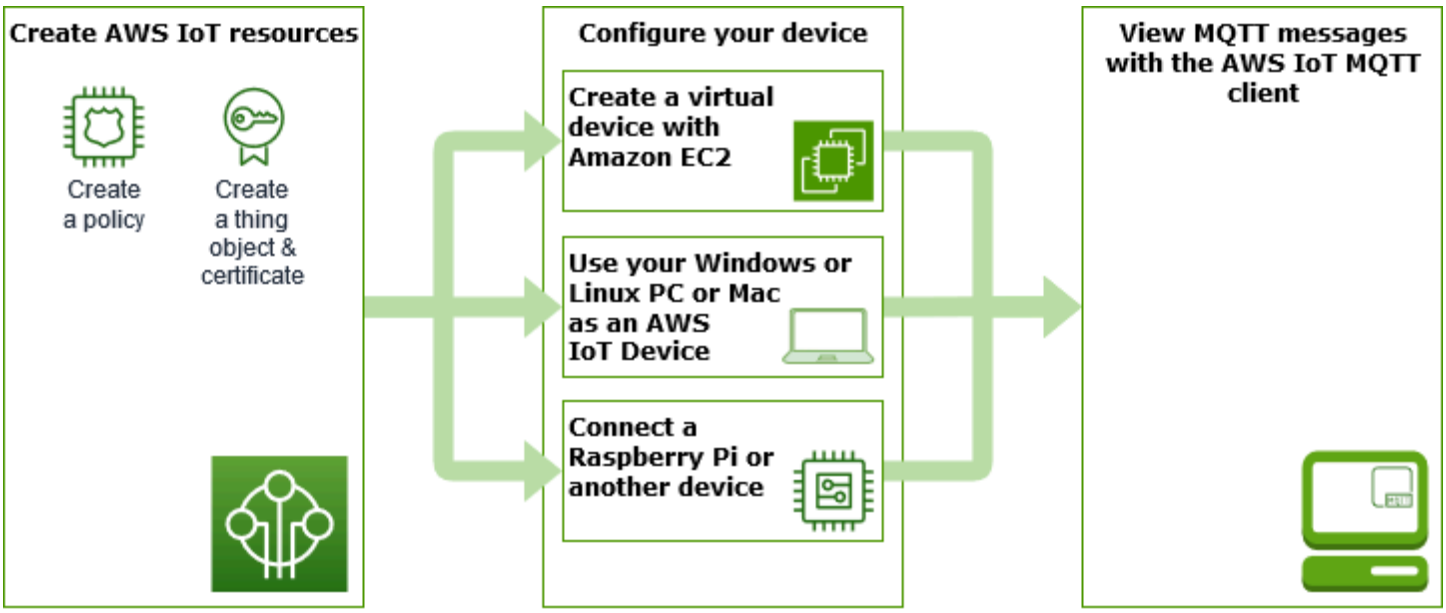
3. Si nmap n'a pas renvoyé de résultat réussi, vérifiez la valeur du point de terminaison pour vous assurer que vous disposez du bon point de terminaison et vérifiez votre connexion de l'appareil avec Internet.

Vous pouvez tester d'autres ports sur le point de terminaison de données de votre appareil, tels que le port 443, le port HTTPS principal, **8443** en remplaçant le port utilisé à l'étape 1 par le port que vous souhaitez tester.

Explorez AWS IoT Core dans des didacticiels pratiques

Dans ce didacticiel, vous allez installer le logiciel et créer les AWS IoT ressources nécessaires pour connecter un appareil AWS IoT Core afin qu'il puisse envoyer et recevoir des messages MQTT avec AWS IoT Core. Vous verrez les messages du client MQTT dans la AWS IoT console.

Attendez à consacrer 20 à 30 minutes à ce didacticiel. Si vous utilisez un appareil IoT ou un Raspberry Pi, ce didacticiel peut prendre plus de temps si, par exemple, vous devez installer le système d'exploitation et configurer l'appareil.



Ce didacticiel est idéal pour les développeurs qui AWS IoT Core souhaitent commencer à explorer des fonctionnalités plus avancées, telles que le moteur de règles et les ombres. Ce didacticiel vous prépare à poursuivre votre apprentissage des autres AWS services AWS IoT Core et de leur interaction avec eux en expliquant les étapes de manière plus détaillée que dans le didacticiel de démarrage rapide. Si vous recherchez juste une expérience rapide, Hello World, essayez le Essayez le didacticiel de connexion AWS IoT Core rapide.

Après avoir configuré votre AWS IoT console Compte AWS et, vous allez suivre les étapes ci-dessous pour savoir comment connecter un appareil et lui demander d'envoyer des messages AWS IoT Core.

Étapes suivantes

- Choisissez l'option d'appareil qui vous convient le mieux
- the section called “Créez des AWS IoT ressources”si vous n'avez pas l'intention de créer un appareil virtuel avec Amazon EC2
- the section called “Configurer votre appareil”
- the section called “Afficher les messages MQTT avec le client AWS IoT MQTT”

Pour plus d'informations AWS IoT Core, voir Qu'est-ce que c'est AWS IoT Core ?

Quelle option d'appareil vous convient le mieux ?

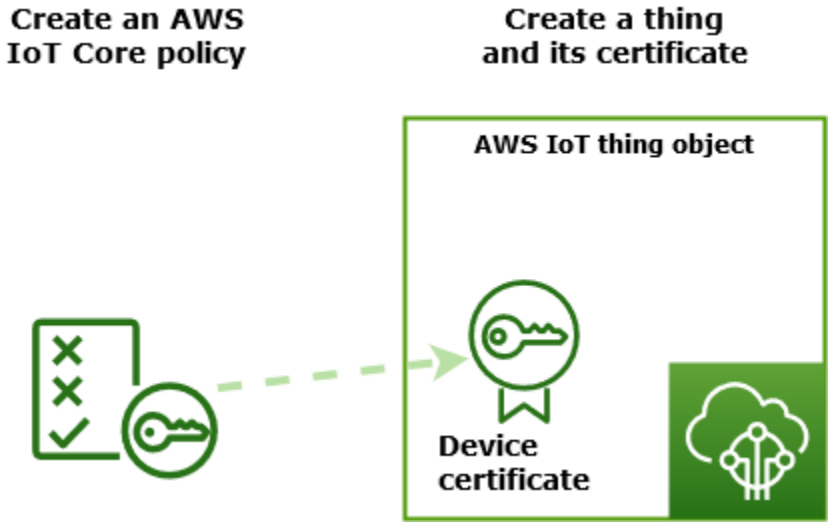
Si vous ne savez pas quelle option choisir, utilisez la liste suivante des avantages et des inconvénients de chaque option pour vous aider à choisir celle qui vous convient le mieux.

Option	Cela peut être une bonne option si :	Cela ne peut être une bonne option si :
the section called “Créez un appareil virtuel avec Amazon EC2”	• Vous n'avez pas d’appareil à tester. • Vous ne souhaitez installer aucun logiciel sur votre système. • Vous souhaitez effectuer un test sur un système d'exploitation Linux.	• Vous n'êtes pas à l'aise avec les commandes de ligne de commande. • Vous ne voulez pas encourir de frais AWS supplémentaires.

Option	Cela peut être une bonne option si :	Cela ne peut être une bonne option si :
		• Vous ne souhaitez effectuer un test sur un système d'exploitation Linux.
the section called “Utilisez votre PC ou Mac Windows ou Linux comme AWS IoT appareil”	• Vous ne voulez pas encourir de frais AWS suppl émentaires. • Vous ne souhaitez pas configurer d'appareils supplémentaires.	• Vous ne souhaitez installer aucun logiciel sur votre ordinateur personnel. • Vous souhaitez une plateforme de test plus représentative.
the section called “Connecte z un Raspberry Pi ou un autre appareil”	• Vous souhaitez effectuer un test AWS IoT avec un appareil réel. • Vous avez déjà un appareil à tester. • Vous avez de l'expérience en matière d'intégration de matériel dans des systèmes.	• Vous ne voulez pas acheter ou configurer un appareil juste pour l'essayer. • Vous voulez tester AWS IoT le plus simplement possible, pour le moment.

Créez des AWS IoT ressources

Dans ce didacticiel, vous allez créer les AWS IoT ressources dont un appareil a besoin pour se connecter AWS IoT Core et échanger des messages.



1. Créez un document AWS IoT de politique qui autorisera votre appareil à interagir avec les AWS IoT services.
2. Créez un objet AWS IoT et son certificat de périphérique X.509, puis joignez le document de politique. L'objet est la représentation virtuelle de votre appareil dans le AWS IoT registre. Le certificat authentifie votre appareil auprès de AWS IoT Core, et le document de politique autorise votre appareil à interagir avec celui-ci. AWS IoT

> **Note**
>
> Si vous avez l'intention de the section called “Créez un appareil virtuel avec Amazon EC2”, vous pouvez ignorer cette page et passer à the section called “Configurer votre appareil”. Vous créerez ces ressources lorsque vous créerez votre objet virtuel.

Ce didacticiel utilise la AWS IoT console pour créer les AWS IoT ressources. Si votre appareil prend en charge un navigateur Web, il peut être plus facile d'exécuter cette procédure sur le navigateur Web de l'appareil, car vous pourrez télécharger les fichiers de certificat directement sur votre appareil. Si vous exécutez cette procédure sur un autre ordinateur, vous devrez copier les fichiers de certificat sur votre appareil avant de pouvoir les utiliser comme exemple d'application.

Création d'une AWS IoT politique

Les appareils utilisent un certificat X.509 pour s'authentifier. AWS IoT Core Le certificat est associé à des AWS IoT politiques. Ces politiques déterminent quelles opérations AWS IoT , telles que

l'abonnement ou la publication sur des sujets MQTT, l'appareil est autorisé à effectuer. Votre appareil présente son certificat lorsqu'il se connecte et envoie des messages à AWS IoT Core.

Suivez les étapes pour créer une politique permettant à votre appareil d'effectuer les opérations AWS IoT nécessaires pour exécuter le programme d'exemple. Vous devez créer la politique AWS IoT avant de pouvoir l'associer au certificat de l'appareil, que vous créerez ultérieurement.

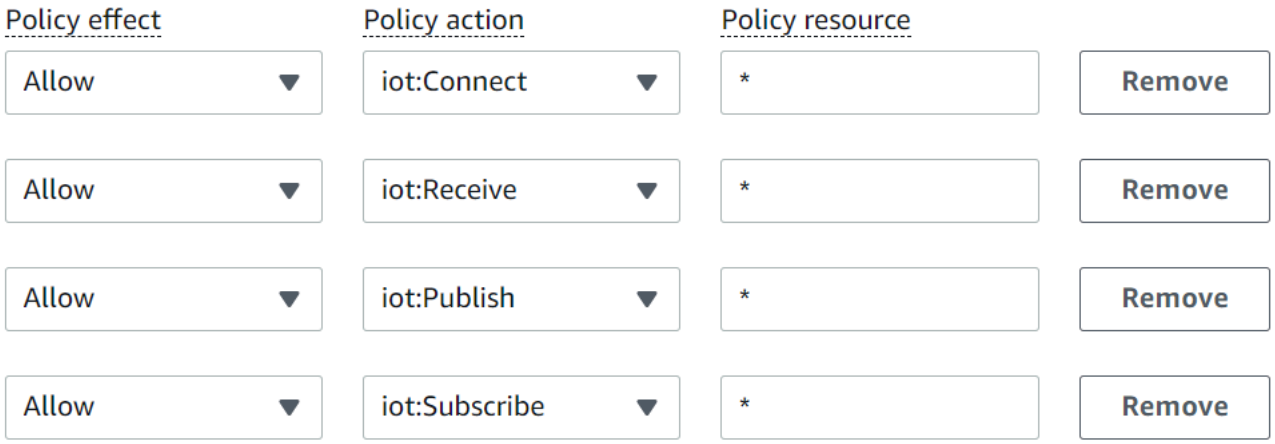
Pour créer une AWS IoT politique

1. Dans la AWS IoT console, dans le menu de gauche, choisissez Sécurité puis choisissez Politiques.
2. Sur la page, Vous n'avez pas encore de stratégie, choisissez Créer une politique.

   Si des politiques existent déjà dans votre compte, choisissez Créer une politique.
3. Sur la page Créer une politique :

   1. Dans la section Propriétés de la politique, dans le champ Nom de la politique, entrez le nom de la politique (par exemple,**`My_Iot_Policy`**). N'utilisez pas d'informations personnelles identifiables dans vos noms de stratégie.
   2. Dans la section Document de politique, créez les déclarations de politique qui accordent ou refusent aux ressources l'accès aux opérations AWS IoT Core . Pour créer une déclaration de politique qui autorise tous les clients à exécuter **`iot:Connect`**, procédez comme suit :
      - Dans le champ Effet de la politique, sélectionnez Autoriser. Cela permet à tous les clients dont cette politique est attachée à leur certificat d'exécuter l'action répertoriée dans le champ Action de la politique.
      - Dans le champ Action de la politique, choisissez une action de politique telle que **`iot:Connect`**. Les actions de politique sont les actions que votre appareil doit être autorisé à effectuer lorsqu'il exécute l'exemple de programme à partir du SDK de l’appareil.
      - Dans le champ Ressource de la politique, entrez une ressource Amazon Resource Name (ARN) ou **`*`**. Un **`*`** pour sélectionner n'importe quel client (appareil).

      Pour créer les déclarations de politique pour **`iot:Receive`**, **`iot:Publish`**, et **`iot:Subscribe`**, choisissez Ajouter une nouvelle déclaration et répétez les étapes.



> **Note**
>
> Dans ce démarrage rapide, le caractère générique (*) est utilisé pour des raisons de simplicité. Pour une sécurité accrue, vous devez limiter les clients (appareils) autorisés à se connecter et à publier des messages en spécifiant un ARN client au lieu du caractère générique comme ressource. Le client ARNs suit ce format :`arn:aws:iot:`*`your-region`*`:`*`your-aws-account`*`:client/`*`my-client-id`*.
>
> Cependant, vous devez d'abord créer la ressource (telle qu'un appareil client ou un objet fantôme) avant de pouvoir attribuer son ARN à une politique. Pour plus d'informations, consultez AWS IoT Core les ressources d'action.

4. Une fois que vous avez entré les informations relatives à votre politique, choisissez Créer.

Pour de plus amples informations, veuillez consulter Comment AWS IoT fonctionne avec IAM.

Créez un objet

Les appareils connectés AWS IoT Core sont représentés par des objets dans le AWS IoT registre. Un objet représente un appareil spécifique ou une entité logique. Il peut s'agir d'un appareil physique ou d'un capteur (par exemple, une ampoule ou un interrupteur mural). Il peut également s'agir d'une entité logique, telle qu'une instance d'une application ou d'une entité physique qui ne se connecte pas AWS IoT, mais qui est associée à d'autres appareils qui le font (par exemple, une voiture équipée de capteurs de moteur ou d'un panneau de commande).

Pour créer un objet dans la AWS IoT console

1. Dans le menu de gauche de la AWS IoT console, choisissez Tous les appareils, puis Objets.
2. Sur la page Objets, choisissez Créer des objets.
3. Sur la page Création d'objets , choisissez Créer un objet unique, puis sur Suivant.
4. Sur la page Spécifier les propriétés de l'objet, dans Nom de l'objet, entrez un nom pour votre objet, tel que **`MyIotThing`**.

   Choisissez les noms des objets avec soin, car vous ne pourrez pas modifier le nom d'un objet ultérieurement.

   Pour changer le nom d'un objet, vous devez créer un objet, lui donner un nouveau nom, puis supprimer l'ancien objet.

   > **Note**
   >
   > N'utilisez pas d'informations personnellement identifiables dans le nom de votre objet. Le nom de l'objet peut apparaître dans les communications et les rapports non chiffrés.

5. Laissez les autres champs de cette page vides. Choisissez Suivant.
6. Sur la page Configurer le certificat de l'appareil - facultatif, choisissez Générer automatiquement un nouveau certificat (recommandé). Choisissez Suivant.
7. Sur la page Attacher des politiques au certificat - facultatif, sélectionnez la politique que vous avez créée dans la section précédente. Dans cette section, la politique était nommée, **`My_Iot_Policy`**. Choisissez Créer objet.
8. Sur la page Télécharger les certificats et les clés :

   1. Téléchargez chacun des fichiers de certificat et de clé et enregistrez-les pour plus tard. Vous devez installer ces fichiers sur votre appareil.

      Lorsque vous enregistrez vos fichiers de certificats, donnez-leur les noms indiqués dans le tableau suivant. Il s'agit des noms de fichiers utilisés dans les exemples suivants.

      Noms des fichiers de certificat

Fichier	Chemin d'accès du fichier
Clé privée	`private.pem.key`

Fichier	Chemin d'accès du fichier
Clé publique	(non utilisé dans ces exemples)
Certificat de l'appareil	`device.pem.crt`
Certificat racine de l'autorité de certification	`Amazon-root-CA-1.pem`

2. Pour télécharger le fichier de l'autorité de certification racine correspondant à ces fichiers, cliquez sur le lien de téléchargement du fichier correspondant au type de point de terminaison de données et de suite de chiffrement que vous utilisez. Dans ce didacticiel, choisissez Télécharger à droite de la clé RSA 2048 bits : Amazon Root CA 1 et téléchargez la clé RSA 2048 bits : fichier de certificat Amazon Root CA 1.

   > **Important**
   >
   > Vous devez enregistrer les fichiers de certificat avant de quitter cette page. Une fois que vous aurez quitté cette page dans la console, vous n'aurez plus accès aux fichiers de certificats.
   > Si vous avez oublié de télécharger les fichiers de certificat que vous avez créés lors de cette étape, vous devez quitter cet écran de console, accéder à la liste des éléments de la console, supprimer l'objet que vous avez créé, puis recommencer cette procédure depuis le début.

3. Sélectionnez Exécuté.

Une fois cette procédure terminée, le nouvel objet devrait apparaître dans votre liste d'objets.

Configurer votre appareil

Cette section décrit comment configurer votre appareil pour qu'il se connecte à AWS IoT Core. Si vous souhaitez commencer AWS IoT Core mais que vous n'avez pas encore d'appareil, vous pouvez créer un appareil virtuel à l'aide d'Amazon EC2 ou utiliser votre PC ou Mac Windows comme appareil IoT.

Sélectionnez l'option d'appareil qui vous convient le mieux AWS IoT Core. Bien sûr, vous pouvez tous les essayer, mais n'en essayez qu'un à la fois. Si vous ne savez pas quelle option d'appareil vous convient le mieux, découvrez comment choisir la meilleure option d'appareil, puis revenez à cette page.

Options de l'appareil

- Créez un appareil virtuel avec Amazon EC2
- Utilisez votre PC ou Mac Windows ou Linux comme AWS IoT appareil
- Connectez un Raspberry Pi ou un autre appareil

Créez un appareil virtuel avec Amazon EC2

Dans ce didacticiel, vous allez créer une EC2 instance Amazon qui servira d'appareil virtuel dans le cloud.

Pour terminer ce didacticiel, vous avez besoin d'un Compte AWS. Si vous n'en avez pas, effectuez les étapes décrites dans la section Configurez Compte AWS avant de continuer.

Dans ce tutoriel, vous allez :

- Configuration d'une EC2 instance Amazon
- Installez Git, Node.js et configurez le AWS CLI
- Créez AWS IoT des ressources pour votre appareil virtuel
- Installez le SDK du AWS IoT périphérique pour JavaScript
- Exécuter les exemples d'applications
- Afficher les messages de l'exemple d'application dans la AWS IoT console

Configuration d'une EC2 instance Amazon

Les étapes suivantes vous montrent comment créer une EC2 instance Amazon qui fera office d'appareil virtuel à la place d'un appareil physique.

Si c'est la première fois que vous créez une EC2 instance Amazon, vous trouverez peut-être les instructions de la section Commencer avec les instances Amazon EC2 Linux plus utiles.

Pour lancer une instance

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le menu de la console sur la gauche, développez la section Instances et choisissez Instances. Dans le tableau de bord Instances, choisissez Lancer les instances sur la droite pour afficher une liste des configurations de base.

3. Dans la section Nom et balises, entrez le nom de l'instance et ajoutez éventuellement des balises.
4. Dans la section Images d'applications et de systèmes d'exploitation (Amazon Machine Image), choisissez un modèle d'AMI pour votre instance, tel que l'AMI Amazon Linux 2 (HVM). Notez que cette AMI est indiquée comme « Éligible à l'offre gratuite ».
5. Dans la section Type d'instance, vous pouvez sélectionner la configuration matérielle de votre instance. Sélectionnez le type `t2.micro` qui est sélectionné par défaut. Notez que ce type d'instance est éligible pour l'offre gratuite.
6. Dans la section Key pair (Paire de clés [login]), choisissez un nom de paire de clés dans la liste déroulante ou choisissez Create a new key pair (Créer une paire de clés) pour en créer une. Lorsque vous créez une nouvelle paire de clés, assurez-vous de télécharger le fichier de clé privée et de l'enregistrer dans un endroit sûr, car c'est votre seule chance de le télécharger et de l'enregistrer. Vous devez fournir le nom de votre paire de clés quand vous lancez une instance, ainsi que la clé privée correspondante chaque fois que vous vous connectez à l'instance.

   > ⚠ Warning
   >
   > Ne choisissez pas l'option Continuer sans paire de clés. Si vous lancez votre instance sans une paire de clés, vous ne pourrez pas vous y connecter.

7. Dans les sections Paramètres réseau et Configuration du stockage, vous pouvez conserver les paramètres par défaut. Une fois que vous êtes prêt, choisissez Lancer les instances.
8. Une page de confirmation indique que l'instance est en cours de lancement. Sélectionnez View Instances pour fermer la page de confirmation et revenir à la console.
9. Sur l'écran Instances, vous pouvez afficher le statut du lancement. Il suffit de peu de temps pour lancer une instance. Lorsque vous lancez une instance, son état initial est `pending`. Une fois que l'instance a démarré, son état devient `running` et elle reçoit un nom DNS public. (Si la colonne DNS public (IPv4) est masquée, choisissez Afficher/Masquer les colonnes (icône en forme d'engrenage) dans le coin supérieur droit de la page, puis sélectionnez DNS public ().) IPv4
10. Cela peut prendre quelques minutes avant que l'instance soit prête pour que vous puissiez vous y connecter. Vérifiez que votre instance a réussi ses contrôles de statut ; vous pouvez voir cette information dans la colonne Status Checks.

    Une fois que le statut de votre nouvelle instance a été vérifié, passez à la procédure suivante et connectez-vous à celle-ci.

Pour vous connecter à votre instance

Vous pouvez vous connecter à une instance à l'aide du client basé sur un navigateur en sélectionnant l'instance depuis la EC2 console Amazon et en choisissant de vous connecter à l'aide d'Amazon Instance EC2 Connect. Instance Connect gère les autorisations et fournit une connexion réussie.

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le menu de gauche, choisissez Instances.
3. Sélectionnez l'instance, puis choisissez Connect (Connexion).
4. Choisissez Amazon EC2 Instance Connect, Connect.

Vous devriez maintenant avoir une fenêtre Amazon EC2 Instance Connect connectée à votre nouvelle EC2 instance Amazon.

Installez Git, Node.js et configurez le AWS CLI

Dans cette section, vous allez installer Git et Node.js sur votre instance Linux.

Pour installer Git

1. Dans votre fenêtre Amazon EC2 Instance Connect, mettez à jour votre instance à l'aide de la commande suivante.

   ```
   sudo yum update -y
   ```

2. Dans votre fenêtre Amazon EC2 Instance Connect, installez Git à l'aide de la commande suivante.

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

3. Pour vérifier si Git a été installé et si la version actuelle de Git est installée, exécutez la commande suivante :

   ```
   git --version
   ```

Pour installer Node.js

1. Dans votre fenêtre Amazon EC2 Instance Connect, installez le gestionnaire de version de nœud (nvm) à l'aide de la commande suivante.

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

   Nous allons utiliser nvm pour installer Node.js, car il peut installer plusieurs versions de Node.js et vous permettre de passer de l'une à l'autre.

2. Dans votre fenêtre Amazon EC2 Instance Connect, activez nvm à l'aide de cette commande.

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

3. Dans votre fenêtre Amazon EC2 Instance Connect, utilisez nvm pour installer la dernière version de Node.js à l'aide de cette commande.

   ```
   nvm install 16
   ```

   > **Note**
   >
   > Cela installe la dernière version LTS de Node.js.

   L'installation de Node.js installe également le gestionnaire de package de nœud (npm), ce qui vous permet d'installer des modules supplémentaires si besoin.

4. Dans votre fenêtre Amazon EC2 Instance Connect, vérifiez que Node.js est installé et s'exécute correctement à l'aide de cette commande.

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

   Ce didacticiel nécessite Nœud v10.0 ou une version ultérieure. Pour plus d'informations, consultez Tutoriel : Configuration de Node.js sur une EC2 instance Amazon.

Pour configurer AWS CLI

Votre EC2 instance Amazon est préchargée avec le AWS CLI. Cependant, vous devez compléter votre AWS CLI profil. Pour plus d'informations sur la configuration de votre CLI, consultez Configuration du AWS CLI.

1. L'exemple suivant montre des exemples de valeurs. Remplacez les par vos propres valeurs. Vous pouvez trouver ces valeurs dans votre AWS console, dans les informations de votre compte sous Identifiants de sécurité.

   Dans votre fenêtre Amazon EC2 Instance Connect, entrez cette commande :

   ```
   aws configure
   ```

   Entrez ensuite les valeurs de votre compte en suivant les instructions affichées.

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

2. Vous pouvez tester votre AWS CLI configuration à l'aide de cette commande :

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

   Si vous êtes correctement AWS CLI configuré, la commande doit renvoyer une adresse de point de terminaison provenant de votre Compte AWS.

Créez AWS IoT des ressources pour votre appareil virtuel

Cette section décrit comment utiliser le AWS CLI pour créer l'objet objet et ses fichiers de certificat directement sur le périphérique virtuel. Cela se fait directement sur l'appareil afin d'éviter les complications potentielles qui pourraient découler de leur copie à partir d'un autre ordinateur. Dans cette section, vous allez créer les ressources suivantes pour votre appareil virtuel :

- Un objet dans lequel représenter votre appareil virtuel AWS IoT.
- Un certificat pour authentifier votre appareil virtuel.
- Document de politique autorisant votre appareil virtuel à se connecter aux messages AWS IoT, à les publier, à les recevoir et à s'y abonner.

Pour créer un AWS IoT objet dans votre instance Linux

Les appareils connectés AWS IoT sont représentés par des objets dans le AWS IoT registre. Un objet représente un appareil spécifique ou une entité logique. Dans ce cas, votre objet objet représentera votre appareil virtuel, cette EC2 instance Amazon.

1. Dans votre fenêtre Amazon EC2 Instance Connect, exécutez la commande suivante pour créer votre objet objet.

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

2. La réponse JSON devrait ressembler à ceci :

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

Pour créer et joindre AWS IoT des clés et des certificats dans votre instance Linux

La commande create-keys-and-certificate crée des certificats clients signés par l'autorité de certification Amazon Root. Ce certificat est utilisé pour authentifier l'identité de votre appareil virtuel.

1. Dans votre fenêtre Amazon EC2 Instance Connect, créez un répertoire pour stocker vos fichiers de certificat et de clé.

```
mkdir ~/certs
```

2. Dans votre fenêtre Amazon EC2 Instance Connect, téléchargez une copie du certificat de l'autorité de certification Amazon (CA) à l'aide de cette commande.

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

3. Dans votre fenêtre Amazon EC2 Instance Connect, exécutez la commande suivante pour créer vos fichiers de clé privée, de clé publique et de certificat X.509. Cette commande enregistre et active également le certificat avec AWS IoT.

```
aws iot create-keys-and-certificate \
```

```
    --set-as-active \
    --certificate-pem-outfile "~/certs/device.pem.crt" \
    --public-key-outfile "~/certs/public.pem.key" \
    --private-key-outfile "~/certs/private.pem.key"
```

La réponse se présente comme suit. Enregistrez le `certificateArn` afin de pouvoir l'utiliser dans les commandes suivantes. Vous en aurez besoin pour joindre votre certificat à votre objet et pour associer la politique au certificat ultérieurement.

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC
 KEY-----\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h
\nMMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/
gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB\n-----
END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
```

```
}
```

4. Dans votre fenêtre Amazon EC2 Instance Connect, attachez votre objet au certificat que vous venez de créer à l'aide de la *certificateArn* commande suivante et de la réponse de la commande précédente.

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

En cas de succès, cette commande n'affiche aucune sortie.

Pour créer et attacher une politique

1. Dans votre fenêtre Amazon EC2 Instance Connect, créez le fichier de politique en copiant et en collant ce document de politique dans un fichier nommé**~/policy.json**.

Si vous n'en avez pas, vous pouvez ouvrir nano à l'aide de cette commande.

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

Collez-y le document de politique pour `policy.json`. Faites ctrl-x pour quitter l'éditeur nano et enregistrer le fichier.

Contenu du document de politique pour `policy.json`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:Connect"
            ],
            "Resource": [
```

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

2. Dans votre fenêtre Amazon EC2 Instance Connect, créez votre politique à l'aide de la commande suivante.

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

Sortie :

JSON

```
{
    "policyName": "MyIotThingPolicy",
    "policyArn": "arn:aws:iot:your-region:your-aws-account:policy/
MyIotThingPolicy",
    "policyDocument": "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [
            {
                \"Effect\": \"Allow\",
                \"Action\": [
                    \"iot:Publish\",
                    \"iot:Receive\",
                    \"iot:Subscribe\",
                    \"iot:Connect\"
                ],
                \"Resource\": [
                    \"*\"
                ]
            }
        ]
    }",
    "policyVersionId": "1"
}
```

3. Dans votre fenêtre Amazon EC2 Instance Connect, associez la politique au certificat de votre appareil virtuel à l'aide de la commande suivante.

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

En cas de succès, cette commande n'affiche aucune sortie.

Installez le SDK du AWS IoT périphérique pour JavaScript

Dans cette section, vous allez installer le SDK du AWS IoT périphérique JavaScript, qui contient le code avec lequel les applications peuvent communiquer, AWS IoT ainsi que des exemples de programmes. Pour plus d'informations, consultez le SDK du AWS IoT périphérique pour le JavaScript GitHub référentiel.

Pour installer le SDK du AWS IoT périphérique JavaScript sur votre instance Linux

1. Dans votre fenêtre Amazon EC2 Instance Connect, clonez le SDK du AWS IoT périphérique pour le JavaScript référentiel dans le `aws-iot-device-sdk-js-v2` répertoire de votre répertoire personnel à l'aide de cette commande.

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

2. Accédez au répertoire `aws-iot-device-sdk-js-v2` que vous avez créé à l'étape précédente.

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

3. Utilisez npm pour installer le kit SDK.

```
npm install
```

Exécuter les exemples d'applications

Les commandes des sections suivantes supposent que vos fichiers de clé et de certificat sont stockés sur votre appareil virtuel comme indiqué dans ce tableau.

Noms des fichiers de certificat

Fichier	Chemin d'accès du fichier
Clé privée	`~/certs/private.pem.key`
Certificat de l'appareil	`~/certs/device.pem.crt`
Certificat racine de l'autorité de certification	`~/certs/Amazon-root-CA-1.pem`

Dans cette section, vous allez installer et exécuter l'`pub-sub.js`exemple d'application qui se trouve dans le `aws-iot-device-sdk-js-v2/samples/node` répertoire du SDK du AWS IoT périphérique pour JavaScript. Cette application montre comment un appareil, votre EC2 instance Amazon, utilise la bibliothèque MQTT pour publier des messages MQTT et s'y abonner. L'exemple d'application `pub-sub.js` s'abonne à une rubrique, `topic_1`, publie 10 messages sur cette rubrique et affiche les messages au fur et à mesure qu'ils sont reçus de l'agent de messages.

Pour installer et exécuter l'exemple d'application

1. Dans votre fenêtre Amazon EC2 Instance Connect, accédez au `aws-iot-device-sdk-js-v2/samples/node/pub_sub` répertoire créé par le SDK et installez l'exemple d'application à l'aide de ces commandes.

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

2. Dans votre fenêtre Amazon EC2 Instance Connect, accédez *`your-iot-endpoint`* à partir de à AWS IoT l'aide de cette commande.

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

3. Dans votre fenêtre Amazon EC2 Instance Connect, insérez *`your-iot-endpoint`* comme indiqué et exécutez cette commande.

   ```
   node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
   certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
   ```

Exemple d'application :

1. Se connecte AWS IoT Core à votre compte.
2. S'abonne à la rubrique du message, topic_1, et affiche les messages qu'il reçoit à cette rubrique.
3. Publie 10 messages dans la rubrique topic_1.
4. Affiche une sortie similaire à celle-ci :

```
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":1}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":2}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":3}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":4}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":5}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":6}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":7}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":8}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":9}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":10}
```

Si vous rencontrez des difficultés en exécutant l'exemple d'application, veuillez consulter the section called “Résoudre les problèmes liés à l'exemple d'application”.

Vous pouvez également ajouter le paramètre `--verbosity debug` sur la ligne de commande afin que l'exemple d'application affiche des messages détaillés sur ce qu'elle fait. Ces informations peuvent vous fournir l'aide dont vous avez besoin pour corriger le problème.

Afficher les messages de l'exemple d'application dans la AWS IoT console

Vous pouvez voir les messages de l'application d'exemple lorsqu'ils passent par l’agent de messages en utilisant le client de test MQTT dans la AWS IoT console.

Pour afficher les messages MQTT publiés par l'exemple d'application

1. Consultez Afficher les messages MQTT avec le client AWS IoT MQTT. Cela vous permet d'apprendre à utiliser le client de test MQTT dans la AWS IoT console pour afficher les messages MQTT lorsqu'ils transitent par l'agent de messages.
2. Ouvrez le client de test MQTT dans la AWS IoT console.
3. Dans S'abonner à une rubrique, S'abonner à la rubrique, topic_1.
4. Dans votre fenêtre Amazon EC2 Instance Connect, réexécutez l'exemple d'application et regardez les messages du client de test MQTT dans la AWS IoT console.

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

Pour plus d'informations sur le MQTT et sur la manière dont le protocole est pris AWS IoT Core en charge, consultez MQTT.

Utilisez votre PC ou Mac Windows ou Linux comme AWS IoT appareil

Dans ce didacticiel, vous allez configurer un ordinateur personnel à utiliser avec AWS IoT. Ces instructions sont compatibles avec Windows, Linux PCs et Mac. Pour ce faire, vous devez installer des logiciels sur votre ordinateur. Si vous ne souhaitez pas installer de logiciel sur votre ordinateur, vous pouvez essayer Créez un appareil virtuel avec Amazon EC2, ce qui permet d'installer tous les logiciels sur une machine virtuelle.

Dans ce tutoriel, vous apprendrez à :

- Configuration de votre ordinateur personnel
- Installation de Git, Python et du SDK AWS IoT Device pour Python
- Configuration de la politique et exécution de l'exemple d'application
- Afficher les messages de l'exemple d'application dans la AWS IoT console
- Exécutez l'exemple d'abonnement partagé dans Python

Configuration de votre ordinateur personnel

Pour suivre ce didacticiel, vous devez disposer d'un PC Windows ou Linux ou d'un Mac connecté à Internet.

Avant de passer à l'étape suivante, assurez-vous de pouvoir ouvrir une fenêtre de ligne de commande sur votre ordinateur. Utiliser cmd.exe sur un PC Windows. Sur un PC Linux ou un Mac, utilisez Terminal.

Installation de Git, Python et du SDK AWS IoT Device pour Python

Dans cette section, vous allez installer Python et le AWS IoT Device SDK for Python sur votre ordinateur.

Installez la dernière version de Git et Python

Cette procédure explique comment installer la dernière version de Git et Python sur votre ordinateur personnel.

Pour télécharger et installer Git et Python sur votre ordinateur

1. Vérifier si Git est installé sur votre ordinateur. Entrez cette commande dans la ligne de commande.

   ```
   git --version
   ```

   Si la commande affiche la version de Git, Git est installé et vous pouvez passer à l'étape suivante.

   Si la commande affiche une erreur, ouvrez https://git-scm.com/downloadet installez Git sur votre ordinateur.
2. Vérifiez si vous avez déjà installé Python. Entrez la commande dans la ligne de commande.

   ```
   python -V
   ```

   > **Note**
   >
   > Si cette commande renvoie une erreur : `Python was not found`, cela peut être dû au fait que votre système d'exploitation appelle l'exécutable Python v3.x en tant que `Python3`. Dans ce cas, remplacez toutes les instances de `python` par `python3` et poursuivez le reste de ce didacticiel.

Si la commande affiche la version de Python, c'est que Python est déjà installé. Ce didacticiel nécessite Python v3.7 ou version ultérieure.

3. Si Python est installé, vous pouvez ignorer le reste des étapes de cette section. Si ce n'est pas le cas, continuez.
4. Ouvrez https://www.python.org/downloads/ et téléchargez le programme d'installation pour votre ordinateur.
5. Si le téléchargement ne démarre pas automatiquement, lancez le programme téléchargé pour installer Python.
6. Vérifier l'installation de Python.

   ```
   python -V
   ```

   Vérifiez que la commande affiche la version de Python. Si la version de Python n'est pas affichée, réessayez de télécharger et d'installer Python.

Installation du SDK du AWS IoT périphérique pour Python

Pour installer le AWS IoT Device SDK pour Python sur votre ordinateur

1. Installez la version 2 du SDK AWS IoT Device pour Python.

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

2. Clonez le référentiel AWS IoT Device SDK for Python dans le répertoire aws-iot-device-sdk - python-v2 de votre répertoire personnel. Cette procédure fait référence au répertoire de base des fichiers sous lesquels vous effectuez l'installation*home*.

   L'emplacement réel du *home* répertoire dépend de votre système d'exploitation.

   Linux/macOS

   Dans macOS et Linux, le *home* répertoire est~.

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

Windows

Sous Windows, vous pouvez trouver le chemin du *home* répertoire en exécutant cette commande dans la `cmd` fenêtre.

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

> **Note**
>
> Si vous utilisez Windows PowerShell au lieu decmd.exe, utilisez la commande suivante.
>
> ```
> echo $home
> ```

Pour plus d'informations, consultez le GitHub référentiel AWS IoT Device SDK pour Python.

Se préparer à exécuter les exemples d'applications

Pour préparer votre système à exécuter l'exemple d'application

- Créez le répertoire `certs`. Dans le répertoire `certs`, copiez les fichiers de clé privée, de certificat de l'appareil et de certificat d'autorité de certification racine que vous avez enregistrés lorsque vous avez créé et enregistré l'objet dans the section called "Créez des AWS IoT ressources". Les noms de chaque fichier du répertoire de destination doivent correspondre à ceux du tableau.

  Les commandes figurant dans la section suivante supposent que vos fichiers de clé et de certificat sont stockés sur votre appareil, comme indiqué dans ce tableau.

  Linux/macOS

  Exécutez cette commande pour créer le sous-répertoire `certs` que vous utiliserez lorsque vous exécuterez les exemples d'applications.

  ```
  mkdir ~/certs
  ```

Dans le nouveau sous-répertoire, copiez les fichiers vers les chemins de fichiers de destination indiqués dans le tableau suivant.

Noms des fichiers de certificat

Fichier	Chemin d'accès du fichier
Clé privée	`~/certs/private.pem.key`
Certificat de l'appareil	`~/certs/device.pem.crt`
Certificat racine de l'autorité de certification	`~/certs/Amazon-root-CA-1.pem`

Exécutez cette commande pour répertorier les fichiers du répertoire `certs` et les comparer à ceux répertoriés dans le tableau.

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

Windows

Exécutez cette commande pour créer le sous-répertoire `certs` que vous utiliserez lorsque vous exécuterez les exemples d'applications.

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

Dans le nouveau sous-répertoire, copiez les fichiers vers les chemins de fichiers de destination indiqués dans le tableau suivant.

Noms des fichiers de certificat

Fichier	Chemin d'accès du fichier
Clé privée	`%USERPROFILE%\certs\private.pem.key`
Certificat de l'appareil	`%USERPROFILE%\certs\device.pem.crt`

Fichier	Chemin d'accès du fichier
Certificat racine de l'autorité de certification	`%USERPROFILE%\certs\Amazon-root-CA-1.pem`

Exécutez cette commande pour répertorier les fichiers du répertoire `certs` et les comparer à ceux répertoriés dans le tableau.

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

Configuration de la politique et exécution de l'exemple d'application

Dans cette section, vous allez configurer votre politique et exécuter l'exemple de script `pubsub.py` qui se trouve dans le `aws-iot-device-sdk-python-v2/samples` répertoire du Kit SDK des appareils AWS IoT pour Python. Ce script montre comment votre appareil utilise la bibliothèque MQTT pour publier et s'abonner aux messages MQTT.

L'exemple d'application `pubsub.py` s'abonne à une rubrique, `test/topic`, publie 10 messages sur cette rubrique et affiche les messages au fur et à mesure qu'ils sont reçus de l’agent de messages.

Pour exécuter l'exemple de script `pubsub.py`, vous avez besoin des informations suivantes :

Valeurs des paramètres de l'application

Paramètre	Où trouver la valeur
`your-iot-endpoint`	1. Dans le menu de gauche de la AWS IoT console, sélectionnez Paramètres. 2. Sur la page Paramètres, votre point de terminaison est affiché dans la section Point de terminaison des données de l'appareil.

La *`your-iot-endpoint`* valeur a le format suivant :*`endpoint_id`*`-ats.iot.`*`region`*`.amazonaws.com`, par exemple,`a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com`.

Avant d'exécuter le script, assurez-vous que la politique de votre objet autorise l'exemple de script à se connecter, à s'abonner, à publier et à recevoir.

Pour rechercher et consulter le document de politique d'une ressource d'objet

1. Dans la AWS IoT console, dans la liste des Objets, recherchez la ressource d'objets qui représente votre appareil.
2. Cliquez sur le lien Nom de la ressource d'objet qui représente votre appareil pour ouvrir la page des détails de l'objet.
3. Sur la page Détails de l'objet, dans l'onglet Certificats, choisissez le certificat attaché à la ressource d'objet. Il ne doit y avoir qu'un seul certificat dans la liste. S'il y en a plusieurs, choisissez le certificat dont les fichiers sont installés sur votre appareil et auquel vous souhaitez vous connecter AWS IoT Core.

   Sur la page Détails du certificat, dans l'onglet Politiques, choisissez la politique attachée au certificat. Il ne doit y en avoir qu'un. S'il y en a plusieurs, répétez l'étape suivante pour chacune afin de vous assurer qu'au moins une politique accorde l'accès requis.
4. Sur la page d'aperçu de la politique, recherchez l'éditeur JSON et choisissez Modifier le document de politique pour consulter et modifier le document de politique selon les besoins.
5. La politique JSON est affichée dans l'exemple suivant. Dans l'`"Resource"`élément, remplacez *`region:account`* par votre Région AWS et Compte AWS dans chacune des `Resource` valeurs.

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account:topic/test/topic"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account:topicfilter/test/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:region:account:client/test-*"
            ]
        }
    ]
}
```

Linux/macOS

Pour exécuter l'exemple de script sous Linux/MacOs

1. Dans votre fenêtre de ligne de commande, accédez au répertoire `~/aws-iot-device-sdk-python-v2/samples/node/pub_sub` créé par le SDK à l'aide de ces commandes.

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

2. Dans votre fenêtre de ligne de commande, remplacez *`your-iot-endpoint`* comme indiqué et exécutez cette commande.

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

Windows

Pour exécuter l'exemple d'application sur un PC Windows

1. Dans votre fenêtre de ligne de commande, accédez au répertoire `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` créé par le SDK et installez l'exemple d'application à l'aide de ces commandes.

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

2. Dans votre fenêtre de ligne de commande, remplacez *your-iot-endpoint* comme indiqué et exécutez cette commande.

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file %USERPROFILE%
\certs\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key
 %USERPROFILE%\certs\private.pem.key
```

Exemple de script :

1. Se connecte AWS IoT Core à votre compte.
2. S'abonne à la rubrique du message, topic/test, et affiche les messages qu'il reçoit à cette rubrique.
3. Publie 10 messages dans la rubrique topic/test.
4. Affiche une sortie similaire à celle-ci :

```
Connected!
Subscribing to topic 'test/topic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'test/topic': Hello World! [1]
Received message from topic 'test/topic': b'"Hello World! [1]"'
Publishing message to topic 'test/topic': Hello World! [2]
Received message from topic 'test/topic': b'"Hello World! [2]"'
Publishing message to topic 'test/topic': Hello World! [3]
Received message from topic 'test/topic': b'"Hello World! [3]"'
Publishing message to topic 'test/topic': Hello World! [4]
Received message from topic 'test/topic': b'"Hello World! [4]"'
Publishing message to topic 'test/topic': Hello World! [5]
Received message from topic 'test/topic': b'"Hello World! [5]"'
Publishing message to topic 'test/topic': Hello World! [6]
```

```
Received message from topic 'test/topic': b'"Hello World! [6]"'
Publishing message to topic 'test/topic': Hello World! [7]
Received message from topic 'test/topic': b'"Hello World! [7]"'
Publishing message to topic 'test/topic': Hello World! [8]
Received message from topic 'test/topic': b'"Hello World! [8]"'
Publishing message to topic 'test/topic': Hello World! [9]
Received message from topic 'test/topic': b'"Hello World! [9]"'
Publishing message to topic 'test/topic': Hello World! [10]
Received message from topic 'test/topic': b'"Hello World! [10]"'
10 message(s) received.
Disconnecting...
Disconnected!
```

Si vous rencontrez des difficultés en exécutant l'exemple d'application, veuillez consulter the section called “Résoudre les problèmes liés à l'exemple d'application”.

Vous pouvez également ajouter le paramètre `--verbosity Debug` sur la ligne de commande afin que l'exemple d'application affiche des messages détaillés sur ce qu'elle fait. Ces informations peuvent vous aider à corriger le problème.

Afficher les messages de l'exemple d'application dans la AWS IoT console

Vous pouvez voir les messages de l'application d'exemple lorsqu'ils passent par l’agent de messages en utilisant le client de test MQTT dans la AWS IoT console.

Pour afficher les messages MQTT publiés par l'exemple d'application

1. Consultez Afficher les messages MQTT avec le client AWS IoT MQTT. Cela vous permet d'apprendre à utiliser le client de test MQTT dans la AWS IoT console pour afficher les messages MQTT lorsqu'ils transitent par l’agent de messages.
2. Ouvrez le client de test MQTT dans la AWS IoT console.
3. Dans S'abonner à une rubrique, abonnez-vous à la rubrique, test/topic.
4. Dans votre fenêtre de ligne de commande, exécutez à nouveau l'exemple d'application et observez les messages du client MQTT dans la AWS IoT console.

   Linux/macOS

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

```
python3 pubsub.py --topic test/topic --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

Windows

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
python3 pubsub.py --topic test/topic --ca_file %USERPROFILE%\certs\Amazon-root-
CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key %USERPROFILE%\certs
\private.pem.key --endpoint your-iot-endpoint
```

Pour plus d'informations sur le MQTT et sur la manière dont le protocole est pris AWS IoT Core en charge, consultez MQTT.

Exécutez l'exemple d'abonnement partagé dans Python

AWS IoT Core prend en charge les abonnements partagés pour MQTT 3 et MQTT 5. Les abonnements partagés permettent à plusieurs clients de partager un abonnement à une rubrique et un seul client recevra les messages publiés sur cette rubrique selon une distribution aléatoire. Pour utiliser les abonnements partagés, les clients s'abonnent au filtre de rubrique d'un abonnement partagé :`$share/{ShareName}/{TopicFilter}`.

Pour configurer la politique et exécuter l'exemple d'abonnement partagé

1. Pour exécuter l'exemple d'abonnement partagé, vous devez configurer la politique de votre objet, comme indiqué dans Abonnement partagé MQTT 5.
2. Pour exécuter l'exemple d'abonnement partagé, exécutez les commandes suivantes.

   Linux/macOS

   Pour exécuter l'exemple de script sous Linux/MacOs

   1. Dans votre fenêtre de ligne de commande, accédez au répertoire `~/aws-iot-device-sdk-python-v2/samples` créé par le SDK à l'aide de ces commandes.

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

   2. Dans votre fenêtre de ligne de commande, remplacez *your-iot-endpoint* comme indiqué et exécutez cette commande.

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

Windows

Pour exécuter l'exemple d'application sur un PC Windows

1. Dans votre fenêtre de ligne de commande, accédez au répertoire `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` créé par le SDK et installez l'exemple d'application à l'aide de ces commandes.

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

2. Dans votre fenêtre de ligne de commande, remplacez *`your-iot-endpoint`* comme indiqué et exécutez cette commande.

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

> **Note**
>
> Vous pouvez éventuellement spécifier un identifiant de groupe en fonction de vos besoins lorsque vous exécutez l'échantillon (par exemple,`--group_identifier consumer`). Si vous n'en spécifiez pas, `python-sample` est l'identifiant de groupe par défaut.

3. Le résultat de votre ligne de commande peut ressembler à ce qui suit :

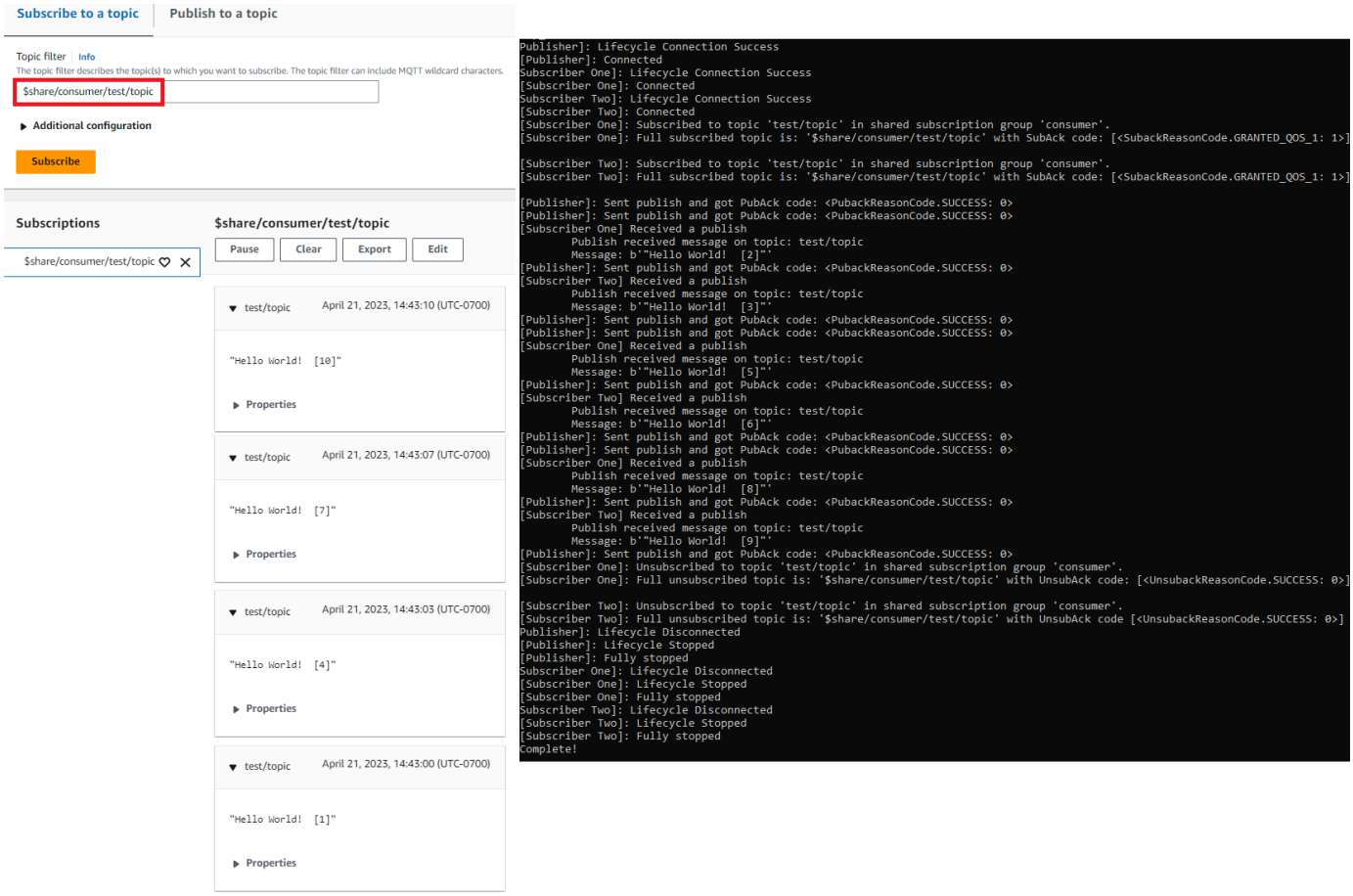
```
Publisher]: Lifecycle Connection Success
[Publisher]: Connected
Subscriber One]: Lifecycle Connection Success
[Subscriber One]: Connected
Subscriber Two]: Lifecycle Connection Success
[Subscriber Two]: Connected
```

```
[Subscriber One]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber One]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Subscriber Two]: Subscribed to topic 'test/topic' in shared subscription group
 'consumer'.
[Subscriber Two]: Full subscribed topic is: '$share/consumer/test/topic' with
 SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [1]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [2]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [3]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [4]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [5]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [6]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [7]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [8]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
        Publish received message on topic: test/topic
        Message: b'"Hello World!  [9]"'
```

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

4. Ouvrez client de test MQTT dans la AWS IoT console. Dans S'abonner à un sujet, abonnez-vous au sujet de l'abonnement partagé tel que :`$share/consumer/test/topic`. Vous pouvez spécifier un identifiant de groupe en fonction de vos besoins lorsque vous exécutez l'échantillon (par exemple,`--group_identifier consumer`). Si vous ne spécifiez pas d'identifiant de groupe, la valeur par défaut est `python-sample`. Pour plus d'informations, consultez l'exemple Python d'abonnement partagé MQTT 5 et le guide AWS IoT Core du développeur sur les abonnements partagés.

   Dans votre fenêtre de ligne de commande, exécutez à nouveau l'exemple d'application et observez la distribution des messages dans votre Client de test MQTT de la AWS IoT console et de la ligne de commande.



Connectez un Raspberry Pi ou un autre appareil

Dans cette section, nous allons configurer un Raspberry Pi à utiliser avec AWS IoT. Si vous souhaitez connecter un autre appareil, les instructions du Raspberry Pi incluent des références qui peuvent vous aider à adapter ces instructions à votre appareil.

Cela prend normalement environ 20 minutes, mais cela peut prendre plus de temps si vous devez installer de nombreuses mises à niveau du logiciel système.

Dans ce tutoriel, vous apprendrez à :

- Configurez votre appareil
- Installez les outils et bibliothèques requis pour le SDK du AWS IoT périphérique
- Installer le SDK AWS IoT du périphérique
- Installez et exécutez l'exemple d'application
- Afficher les messages de l'exemple d'application dans la AWS IoT console

> **Important**
>
> L'adaptation de ces instructions à d'autres appareils et systèmes d'exploitation peut s'avérer difficile. Vous devez avoir une connaissance suffisante de votre appareil pour être en mesure d'interpréter ces instructions et de les appliquer.
> Si vous rencontrez des difficultés lors de la configuration de votre appareil pour AWS IoT, vous pouvez essayer l'une des autres options de l'appareil comme alternative, telle que Créez un appareil virtuel avec Amazon EC2 ouUtilisez votre PC ou Mac Windows ou Linux comme AWS IoT appareil.

Configurez votre appareil

L'objectif de cette étape est de collecter les informations dont vous aurez besoin pour configurer votre appareil afin qu'il puisse démarrer le système d'exploitation (OS), se connecter à Internet et vous permettre d'interagir avec celui-ci via une interface de ligne de commande.

Pour suivre ce didacticiel, vous aurez besoin des éléments suivants :

- Un Compte AWS. Si vous n'en avez pas, effectuez les étapes décrites dans Configurez Compte AWS avant de continuer.
- Un Raspberry Pi 3 modèle B ou un modèle plus récent. Cela peut fonctionner sur les versions antérieures du Raspberry Pi, mais elles n'ont pas été testées.
- Système d'exploitation Raspberry Pi (32 bits) ou version ultérieure. Nous vous recommandons d'utiliser la dernière version du système d'exploitation Raspberry Pi. Les versions antérieures du système d'exploitation peuvent fonctionner, mais elles n'ont pas été testées.

  Pour exécuter cet exemple, vous n'avez pas besoin d'installer le bureau avec l'interface utilisateur graphique (GUI) ; cependant, si vous êtes nouveau sur le Raspberry Pi et que votre matériel Raspberry Pi le prend en charge, l'utilisation du bureau avec l'interface graphique peut être plus facile.
- Une WiFi connexion Ethernet.
- Clavier, souris, écran, câbles, blocs d'alimentation et autres équipements nécessaires à votre appareil.

> **Important**
>
> Avant de passer à l'étape suivante, le système d'exploitation de votre appareil doit être installé, configuré et en cours d'exécution. L'appareil doit être connecté à Internet et vous devez pouvoir y accéder à l'aide de son interface en ligne de commande. L'accès à la ligne de commande peut se faire via un clavier, une souris et un moniteur directement connectés, ou en utilisant une interface distante de terminal SSH.

Si vous exécutez un système d'exploitation sur votre Raspberry Pi doté d'une interface utilisateur graphique (GUI), ouvrez une fenêtre de terminal sur l'appareil et suivez les instructions suivantes dans cette fenêtre. Sinon, si vous vous connectez à votre appareil à l'aide d'un terminal distant, tel que PuTTY, ouvrez un terminal distant sur votre appareil et utilisez-le.

Installez les outils et bibliothèques requis pour le SDK du AWS IoT périphérique

Avant d'installer le SDK du AWS IoT périphérique et un exemple de code, assurez-vous que votre système est à jour et qu'il dispose des outils et bibliothèques nécessaires pour installer le SDKs.

1. Mise à jour du système d'exploitation et installation des bibliothèques requises

   Avant d'installer un SDK de AWS IoT périphérique, exécutez ces commandes dans une fenêtre de terminal de votre appareil pour mettre à jour le système d'exploitation et installer les bibliothèques requises.

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```

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

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

2. Installez Git

   Si Git n'est pas installé sur le système d'exploitation de votre appareil, vous devez l'installer pour installer le SDK du AWS IoT périphérique pour JavaScript.

a. Vérifiez si Git est déjà installé en exécutant cette commande.

```
git --version
```

b. Si la commande précédente renvoie la version de Git, cela signifie que Git est déjà installé et vous pouvez passer à l'étape 3.

c. Si une erreur s'affiche lorsque vous exécutez la commande git, installez Git en exécutant cette commande.

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

d. Testez à nouveau pour voir si Git est installé en exécutant cette commande.

```
git --version
```

e. Si Git est installé, passez à la section suivante. Si ce n'est pas le cas, dépannez et corrigez l'erreur avant de continuer. Vous avez besoin de Git pour installer le AWS IoT Device SDK pour JavaScript.

Installer le SDK AWS IoT du périphérique

Installez le SDK du AWS IoT périphérique.

Python

Dans cette section, vous allez installer Python, ses outils de développement et le AWS IoT Device SDK pour Python sur votre appareil. Ces instructions concernent un Raspberry Pi exécutant le dernier système d’exploitation Raspberry Pi. Si vous avez un autre appareil ou utilisez un autre système d'exploitation, vous devrez peut-être adapter ces instructions à votre appareil.

1. Installer Python et ses outils de développement

   Le AWS IoT Device SDK pour Python nécessite l'installation de Python v3.5 ou version ultérieure sur votre Raspberry Pi.

   Dans une fenêtre de terminal sur votre appareil, exécutez ces commandes.

   1. Exécutez cette commande pour déterminer la version de Python installée sur votre appareil.

```
python3 --version
```

Si Python est installé, il affichera sa version.

2. Si la version affichée est `Python 3.5` ou supérieure, vous pouvez passer à l'étape 2.
3. Si la version affichée est inférieure à `Python 3.5`, vous pouvez installer la bonne version en exécutant cette commande.

```
sudo apt install python3
```

4. Exécutez cette commande pour vérifier que la bonne version de Python est désormais installée.

```
python3 --version
```

2. Test pour pip3

Dans une fenêtre de terminal sur votre appareil, exécutez ces commandes.

1. Exécutez cette commande pour voir si pip3 est installée.

```
pip3 --version
```

2. Si la commande renvoie un numéro de version, pip3 est installée et vous pouvez passer à l'étape 3.
3. Si la commande précédente renvoie une erreur, exécutez cette commande pour effectuer l'installation de pip3.

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

4. Exécutez cette commande pour voir si pip3 est installée.

```
pip3 --version
```

3. Installez le SDK AWS IoT Device actuel pour Python

Installez le SDK AWS IoT Device pour Python et téléchargez les exemples d'applications sur votre appareil.

Sur votre appareil, exécutez ces commandes.

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

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

JavaScript

Dans cette section, vous allez installer Node.js, le gestionnaire de packages npm et le AWS IoT Device SDK for JavaScript sur votre appareil. Ces instructions concernent un Raspberry Pi exécutant le système d'exploitation Raspberry Pi. Si vous avez un autre appareil ou utilisez un autre système d'exploitation, vous devrez peut-être adapter ces instructions à votre appareil.

1. Installer la dernière version de Node.js

   Le SDK du AWS IoT périphérique pour JavaScript nécessite l'installation de Node.js et du gestionnaire de packages npm sur votre Raspberry Pi.

   a. Téléchargez la dernière version du référentiel Nœud en entrant cette commande.

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

   b. Installez Nœud et npm.

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

   c. Vérifier l'installation de Nœud.

   ```
   node -v
   ```

   Confirmez que la commande affiche la version du Nœud. Ce didacticiel nécessite Nœud v10.0 ou une version ultérieure. Si la version de Nœud n'est pas affichée, essayez de télécharger à nouveau le référentiel Nœud.

   d. Vérifiez l'installation de npm.

   ```
   npm -v
   ```

Confirmez que la commande affiche la version du Nœud npm. Si la version npm n'est pas affichée, réessayez d'installer Nœud et npm.

e. Redémarrez le périphérique.

```
sudo shutdown -r 0
```

Continuez après le redémarrage de l'appareil.

2. Installez le SDK du AWS IoT périphérique pour JavaScript

Installez le SDK du AWS IoT périphérique pour JavaScript votre Raspberry Pi.

a. Clonez le SDK du AWS IoT périphérique pour le JavaScript référentiel dans le `aws-iot-device-sdk-js-v2` répertoire de votre *home* répertoire. Sur le Raspberry Pi`~/`, le *home* répertoire est, qui est utilisé comme *home* répertoire dans les commandes suivantes. Si votre appareil utilise un chemin différent pour le *home* répertoire, vous devez le `~/` remplacer par le chemin correct pour votre appareil dans les commandes suivantes.

Ces commandes créent le répertoire `~/aws-iot-device-sdk-js-v2` et y copient le code du SDK.

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

b. Accédez au répertoire `aws-iot-device-sdk-js-v2` que vous avez créé à l'étape précédente et lancez-le `npm install` pour installer le SDK. La commande `npm install` invoque la compilation de la bibliothèque `aws-crt`, qui peut prendre quelques minutes.

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

Installez et exécutez l'exemple d'application

Dans cette section, vous allez installer et exécuter l'`pubsub`exemple d'application qui se trouve dans le SDK de l' AWS IoT appareil. Cette application montre comment votre appareil utilise la bibliothèque MQTT pour publier et s'abonner aux messages MQTT. L'exemple d'application s'abonne

à une rubrique, `topic_1`, publie 10 messages sur cette rubrique et affiche les messages au fur et à mesure qu'ils sont reçus de l'agent de messages.

Installez les fichiers de certificat

L'exemple d'application nécessite que les fichiers de certificat authentifiant l'appareil soient installés sur celui-ci.

Pour installer les fichiers de certificat de l'appareil pour l'exemple d'application

1. Créez un `certs` sous-répertoire dans votre *home* répertoire en exécutant ces commandes.

   ```
   cd ~
   mkdir certs
   ```

2. Dans le répertoire `~/certs`, copiez la clé privée, le certificat de périphérique et le certificat d'autorité de certification racine que vous avez créés précédemment dans the section called "Créez des AWS IoT ressources".

   La façon dont vous copiez les fichiers de certificat sur votre appareil dépend de l'appareil et du système d'exploitation et n'est pas décrite ici. Toutefois, si votre appareil prend en charge une interface utilisateur graphique (GUI) et dispose d'un navigateur Web, vous pouvez suivre la procédure décrite dans le navigateur Web de votre appareil the section called "Créez des AWS IoT ressources" pour télécharger les fichiers obtenus directement sur votre appareil.

   Les commandes figurant dans la section suivante supposent que vos fichiers de clé et de certificat sont stockés sur l'appareil, comme indiqué dans ce tableau.

   Noms des fichiers de certificat

Fichier	Chemin d'accès du fichier
Certificat racine de l'autorité de certification	`~/certs/Amazon-root-CA-1.pem`
Certificat de l'appareil	`~/certs/device.pem.crt`
Clé privée	`~/certs/private.pem.key`

Pour exécuter cet exemple d'application, vous avez besoin des informations suivantes :

Valeurs des paramètres de l'application

Paramètre	Où trouver la valeur
your-iot-endpoint	Dans la AWS IoT console, choisissez Tous les appareils, puis Objets. Sur la page Paramètres du AWS IoT menu. Votre point de terminaison est affiché dans la section Point de terminaison des données de l'appareil.

La *your-iot-endpoint* valeur a le format suivant :*endpoint_id*-ats.iot.*region*.amazonaws.com, par exemple,a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

Python

Pour installer et exécuter l'exemple d'application

1. Accédez au répertoire de l’exemple d’application.

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

2. Dans la fenêtre de ligne de commande, remplacez *your-iot-endpoint* comme indiqué et exécutez cette commande.

```
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Notez que l'exemple d'application :

    1. Se connecte au AWS IoT service associé à votre compte.
    2. S'abonne à la rubrique du message, topic_1, et affiche les messages qu'il reçoit à cette rubrique.
    3. Publie 10 messages dans la rubrique topic_1.
    4. Affiche une sortie similaire à celle-ci :

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to topic 'topic_1'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'topic_1': Hello World! [1]
Received message from topic 'topic_1': b'Hello World! [1]'
Publishing message to topic 'topic_1': Hello World! [2]
Received message from topic 'topic_1': b'Hello World! [2]'
Publishing message to topic 'topic_1': Hello World! [3]
Received message from topic 'topic_1': b'Hello World! [3]'
Publishing message to topic 'topic_1': Hello World! [4]
Received message from topic 'topic_1': b'Hello World! [4]'
Publishing message to topic 'topic_1': Hello World! [5]
Received message from topic 'topic_1': b'Hello World! [5]'
Publishing message to topic 'topic_1': Hello World! [6]
Received message from topic 'topic_1': b'Hello World! [6]'
Publishing message to topic 'topic_1': Hello World! [7]
Received message from topic 'topic_1': b'Hello World! [7]'
Publishing message to topic 'topic_1': Hello World! [8]
Received message from topic 'topic_1': b'Hello World! [8]'
Publishing message to topic 'topic_1': Hello World! [9]
Received message from topic 'topic_1': b'Hello World! [9]'
Publishing message to topic 'topic_1': Hello World! [10]
Received message from topic 'topic_1': b'Hello World! [10]'
10 message(s) received.
Disconnecting...
Disconnected!
```

Si vous rencontrez des difficultés en exécutant l'exemple d'application, veuillez consulter the section called “Résoudre les problèmes liés à l'exemple d'application”.

Vous pouvez également ajouter le paramètre `--verbosity Debug` sur la ligne de commande afin que l'exemple d'application affiche des messages détaillés sur ce qu'elle fait. Ces informations peuvent vous fournir l'aide dont vous avez besoin pour corriger le problème.

JavaScript

Pour installer et exécuter l'exemple d'application

1. Dans votre fenêtre de ligne de commande, accédez au répertoire `~/aws-iot-device-sdk-js-v2/samples/node/pub_sub` créé par le SDK et installez l'exemple d'application à l'aide de ces commandes. La commande `npm install` invoque la compilation de la bibliothèque `aws-crt`, qui peut prendre quelques minutes.

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

2. Dans la fenêtre de ligne de commande, remplacez *`your-iot-endpoint`* comme indiqué et exécutez cette commande.

```
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. Notez que l'exemple d'application :

   1. Se connecte au AWS IoT service associé à votre compte.
   2. S'abonne à la rubrique du message, topic_1, et affiche les messages qu'il reçoit à cette rubrique.
   3. Publie 10 messages dans la rubrique topic_1.
   4. Affiche une sortie similaire à celle-ci :

```
Publish received on topic topic_1
{"message":"Hello world!","sequence":1}
Publish received on topic topic_1
{"message":"Hello world!","sequence":2}
Publish received on topic topic_1
{"message":"Hello world!","sequence":3}
Publish received on topic topic_1
{"message":"Hello world!","sequence":4}
Publish received on topic topic_1
{"message":"Hello world!","sequence":5}
Publish received on topic topic_1
{"message":"Hello world!","sequence":6}
Publish received on topic topic_1
```

```
{"message":"Hello world!","sequence":7}
Publish received on topic topic_1
{"message":"Hello world!","sequence":8}
Publish received on topic topic_1
{"message":"Hello world!","sequence":9}
Publish received on topic topic_1
{"message":"Hello world!","sequence":10}
```

Si vous rencontrez des difficultés en exécutant l'exemple d'application, veuillez consulter the section called “Résoudre les problèmes liés à l'exemple d'application”.

Vous pouvez également ajouter le paramètre `--verbosity Debug` sur la ligne de commande afin que l'exemple d'application affiche des messages détaillés sur ce qu'elle fait. Ces informations peuvent vous fournir l'aide dont vous avez besoin pour corriger le problème.

Afficher les messages de l'exemple d'application dans la AWS IoT console

Vous pouvez voir les messages de l'application d'exemple lorsqu'ils passent par l’agent de messages en utilisant le client de test MQTT dans la AWS IoT console.

Pour afficher les messages MQTT publiés par l'exemple d'application

1. Consultez Afficher les messages MQTT avec le client AWS IoT MQTT. Cela vous permet d'apprendre à utiliser le client de test MQTT dans la AWS IoT console pour afficher les messages MQTT lorsqu'ils transitent par l’agent de messages.
2. Ouvrez le client de test MQTT dans la AWS IoT console.
3. S'abonner à la rubrique, topic_1.
4. Dans votre fenêtre de ligne de commande, exécutez à nouveau l'exemple d'application et observez les messages du client MQTT dans la AWS IoT console.

   Python

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
   cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

JavaScript

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --
cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

Résoudre les problèmes liés à l'exemple d'application

Si vous rencontrez une erreur lorsque vous essayez d'exécuter l'exemple d'application, voici quelques points à vérifier.

Vérifiez le certificat

Si le certificat n'est pas actif, AWS IoT il n'acceptera aucune tentative de connexion l'utilisant à des fins d'autorisation. Lorsque vous créez votre certificat, il est facile d'oublier le bouton Activer. Heureusement, vous pouvez activer votre certificat depuis la AWS IoT console.

Pour vérifier l'activation de votre certificat

1. Dans la AWS IoT console, dans le menu de gauche, choisissez Sécurisé, puis Certificats.
2. Dans la liste des certificats, recherchez le certificat que vous avez créé pour l'exercice et vérifiez son statut dans la colonne État.

   Si vous ne vous souvenez pas du nom du certificat, vérifiez s'il est inactif pour voir s'il s'agit bien de celui que vous utilisez.

   Choisissez le certificat dans la liste pour ouvrir sa page de détail. Sur la page détaillée, vous pouvez voir sa date de création pour vous aider à identifier le certificat.
3. Pour activer un certificat inactif, sur la page détaillée du certificat, sélectionnez Actions, puis sélectionnez Activer.

Si vous avez trouvé le bon certificat et qu'il est actif, mais que vous rencontrez toujours des problèmes lors de l'exécution de l'exemple d'application, vérifiez sa politique comme décrit à l'étape suivante.

Vous pouvez également essayer de créer un nouvel objet et un nouveau certificat en suivant les étapes décrites dans the section called “Créez un objet”. Si vous créez un nouvel objet, vous devrez lui attribuer un nouveau nom et télécharger les nouveaux fichiers de certificat sur votre appareil.

Vérifiez la politique attachée au certificat

Les politiques autorisent les actions dans AWS IoT. Si le certificat utilisé pour se connecter à AWS IoT n'a pas de politique ou n'a pas de politique lui permettant de se connecter, la connexion sera refusée, même si le certificat est actif.

Pour vérifier les politiques attachées à un certificat

1. Recherchez le certificat comme décrit dans l'article précédent et ouvrez sa page de détails.
2. Dans le menu de gauche de la page de détails du certificat, choisissez Politiques pour voir les politiques attachées au certificat.
3. Si aucune politique n'est attachée au certificat, ajoutez-en une en choisissant le menu Actions, puis en choisissant Attacher une politique.

   Choisissez la politique que vous avez créée précédemment dans the section called “Créez des AWS IoT ressources”.
4. Si une politique est jointe, sélectionnez la vignette de politique pour ouvrir sa page de détails.

   Sur la page de détails, consultez le document de politique pour vous assurer qu'il contient les mêmes informations que celles que vous avez créées dans the section called “Création d'une AWS IoT politique”.

Vérifiez la ligne de commande

Assurez-vous d'avoir utilisé la bonne ligne de commande pour votre système. Les commandes utilisées sur les systèmes Linux et MacOs sont souvent différentes de celles utilisées sur les systèmes Windows.

Vérifiez l'adresse du point de terminaison

Passez en revue la commande que vous avez saisie et vérifiez que l'adresse du point de terminaison indiquée dans votre commande correspond à celle de votre AWS IoT console.

Vérifiez les noms des fichiers de certificat

Comparez les noms de fichiers figurant dans la commande que vous avez saisie aux noms de fichiers des certificats du répertoire `certs`.

Certains systèmes peuvent nécessiter que les noms de fichiers soient entre guillemets pour fonctionner correctement.

Vérifiez l'installation du SDK

Assurez-vous que l'installation du SDK est complète et correcte.

En cas de doute, réinstallez le SDK sur votre appareil. Dans la plupart des cas, il suffit de trouver la section du didacticiel intitulée Installer le SDK du AWS IoT périphérique pour ***SDK language*** et de suivre à nouveau la procédure.

Si vous utilisez le SDK du AWS IoT périphérique pour JavaScript, n'oubliez pas d'installer les exemples d'applications avant d'essayer de les exécuter. L'installation du SDK n'installe pas automatiquement les exemples d'applications. Les exemples d'applications doivent être installés manuellement après l'installation du SDK.

Afficher les messages MQTT avec le client AWS IoT MQTT

Cette section décrit comment utiliser le client de test AWS IoT MQTT dans la AWS IoT console pour surveiller les messages MQTT envoyés et reçus par. AWS IoT L'exemple utilisé dans cette section concerne les exemples utilisés dans Commencer à utiliser les AWS IoT Core didacticiels ; toutefois, vous pouvez remplacer le nom *topicName* utilisé dans les exemples par n'importe quel nom de rubrique ou filtre de sujet utilisé par votre solution IoT.

Les appareils publient des messages MQTT identifiés par des sujets auxquels ils doivent communiquer leur état AWS IoT, et AWS IoT publient des messages MQTT pour informer les appareils et les applications des changements et des événements. Vous pouvez utiliser le client MQTT pour vous abonner à ces rubriques et consulter les messages au fur et à mesure qu'ils apparaissent. Vous pouvez également utiliser le client de test MQTT pour publier des messages MQTT sur les appareils et services auxquels vous êtes abonné dans votre. Compte AWS

Table des matières

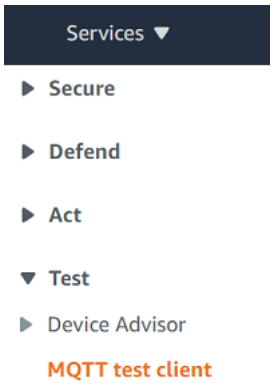
- Affichage des messages MQTT dans le client MQTT
- Publication de messages MQTT à partir du client MQTT
- Test des abonnements partagés dans le client MQTT

Affichage des messages MQTT dans le client MQTT

La procédure suivante explique comment s'abonner à un sujet MQTT spécifique sur lequel votre appareil publie des messages et comment afficher ces messages dans la AWS IoT console.

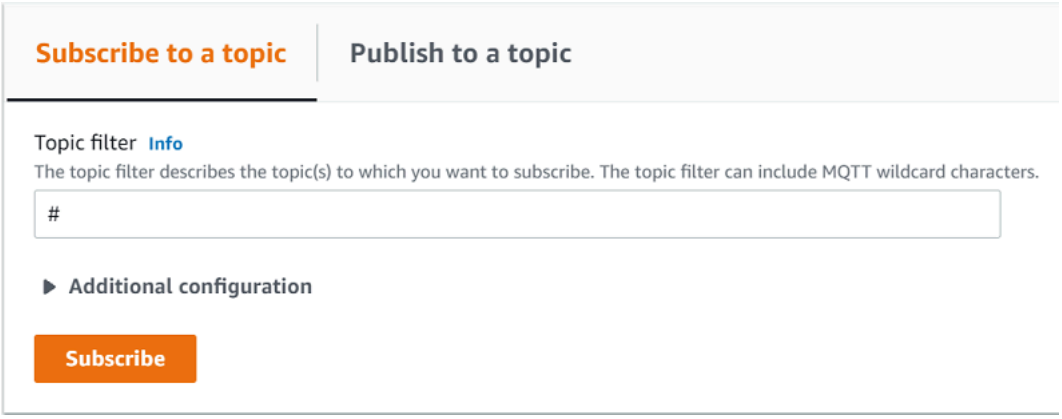
Pour afficher les messages MQTT dans le client MQTT

1. Dans la AWS IoT console, dans le menu de gauche, choisissez Test puis choisissez Client de test MQTT.



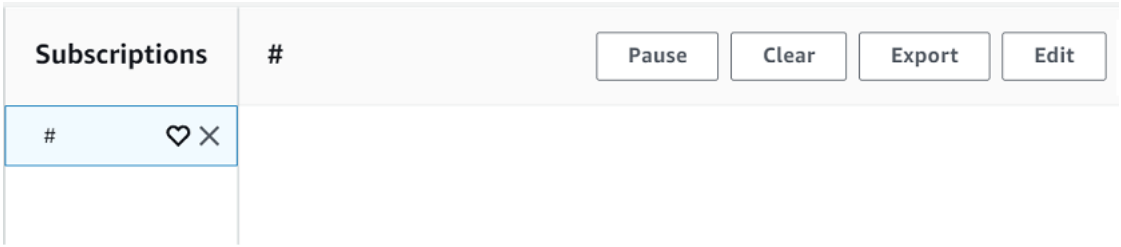
2. Dans l'onglet S'abonner à un sujet, entrez le *topicName* pour vous abonner au sujet sur lequel votre appareil publie. Pour l'exemple d'application de démarrage, abonnez-vous à **#**, qui s'abonne à tous les rubriques de message.

   En reprenant l'exemple de démarrage, sous l'onglet S'abonner à une rubrique, dans le champ Filtre de rubrique, saisissez **#**, puis choisissez S'abonner.



   La page du journal des messages de la rubrique, #, s'ouvre et **#** apparaît dans la liste des abonnements. Si le périphérique que vous avez the section called "Configurer votre appareil" configuré exécute le programme d'exemple, vous devriez voir les messages qu'il envoie AWS IoT dans le journal des messages #. Les entrées du journal des messages apparaîtront sous la

section Publier lorsque les messages contenant le sujet auquel vous êtes abonné seront reçus par AWS IoT.



3. Sur la page du journal des messages #, vous pouvez également publier des messages dans une rubrique, mais vous devez spécifier son nom. Vous ne pouvez pas publier sur la rubrique #.

   Les messages publiés dans les sujets auxquels vous êtes abonné apparaissent dans le journal des messages au fur et à mesure de leur réception, le message le plus récent étant placé en premier.

Résolution des problèmes MQTT

Utiliser le filtre de rubrique générique

Si vos messages n'apparaissent pas dans le journal des messages comme prévu, essayez de vous abonner à un filtre de rubrique générique comme décrit dans Filtres de noms de sujets. Le filtre de rubrique générique multiniveau MQTT est le dièse ou le signe dièse (#) et peut être utilisé comme filtre de sujet dans le champ Sujet d'abonnement.

L'abonnement au # le filtre de rubrique s'abonne à chaque rubrique reçue par l’agent de messages. Vous pouvez affiner le filtre en remplaçant les éléments du chemin du filtre par rubrique par un caractère générique à # de plusieurs niveaux ou le caractère joker '+' à un seul niveau.

Lorsque vous utilisez des caractères génériques dans un filtre de rubrique

- Le caractère générique à plusieurs niveaux doit être le dernier caractère dans le filtre de rubrique.
- Le chemin du filtre de rubrique ne peut comporter qu'un seul caractère générique par niveau de rubrique.

Par exemple :

Filtre de rubriques	Affiche les messages avec
#	Un nom de rubrique quelconque
`topic_1/#`	Un nom de rubrique commençant par `topic_1/`
`topic_1/level_2/#`	Un nom de rubrique commençant par `topic_1/level_2/`
`topic_1/+/level_3`	Nom de rubrique commençant par `topic_1/`, se terminant par `/level_3` et comportant un élément de n'importe quelle valeur entre les deux.

Pour plus d'informations sur les filtres de rubriques, consultez Filtres de noms de sujets.

Vérifiez les erreurs dans le nom de rubrique

Les noms de rubrique MQTT et les filtres de rubrique sont sensibles à la casse. Si, par exemple, votre appareil publie des messages sur `Topic_1` (avec un T majuscule) au lieu de `topic_1`, la rubrique où vous êtes abonné, ses messages n'apparaîtront pas dans le client de test MQTT. L'abonnement au filtre de rubrique générique indiquerait toutefois que l'appareil publie des messages et que vous pourriez voir qu'il utilise un nom de rubrique qui n'est pas celui que vous attendiez.

Publication de messages MQTT à partir du client MQTT

Pour publier un message dans une rubrique MQTT

1. Sur la page du client de test MQTT, dans l'onglet Publier dans un sujet, dans le champ Nom du sujet, entrez le message *`topicName`* de votre message. Pour cet exemple, utilisez **`my/topic`**.

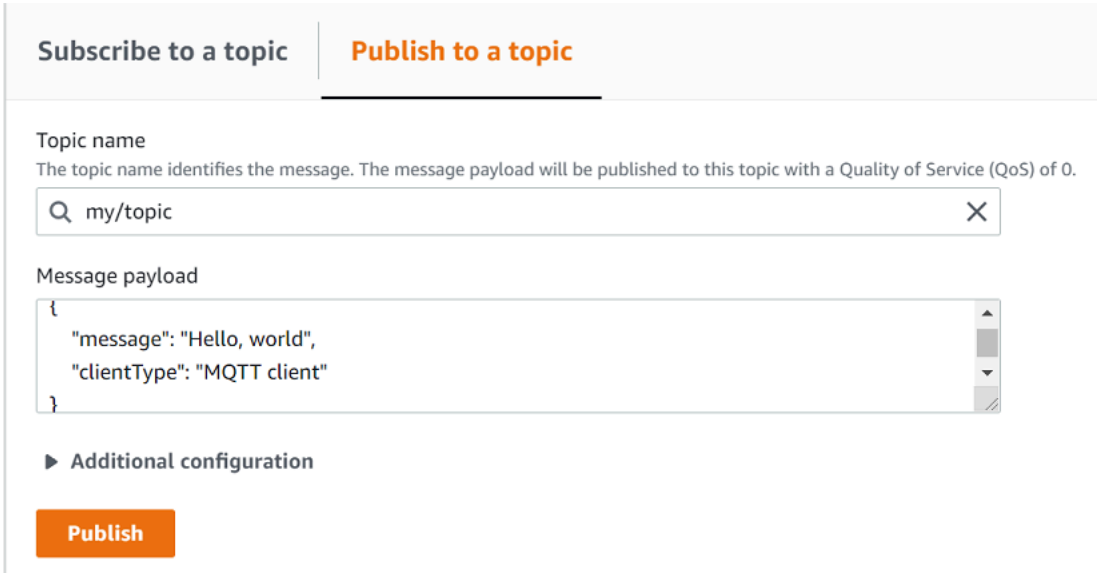
   > **Note**
   >
   > N'utilisez pas d'informations personnelles identifiables dans les noms de rubriques, lorsque vous les utilisiez dans le client MQTT ou dans l'implémentation de votre système. Le nom de rubriques peut apparaître dans les communications et les rapports non chiffrés.

2. Dans la section de charge utile du message, saisissez le contenu JSON suivant :

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

3. Choisissez Publier pour publier votre message dans AWS IoT.

> **Note**
>
> Assurez-vous d'être abonné à la rubrique my/topic avant de publier votre message.



4. Dans la liste Abonnement choisissez my/topic pour afficher le message. Vous devriez voir le message apparaître dans le client de test MQTT sous la fenêtre de charge utile du message de publication.

Vous pouvez publier des messages MQTT sur d'autres sujets *topicName* en modifiant le champ Nom du sujet et en cliquant sur le bouton Publier.

> **Important**
>
> Lorsque vous créez plusieurs abonnements dont les sujets se chevauchent (par exemple, probe1/temperature et probe1/#), il est possible qu'un seul message publié sur un sujet correspondant aux deux abonnements soit diffusé plusieurs fois, une fois pour chaque abonnement qui se chevauche.

Test des abonnements partagés dans le client MQTT

Cette section décrit comment utiliser le client AWS IoT MQTT dans la AWS IoT console pour surveiller les messages MQTT envoyés et reçus à AWS IoT l'aide d'abonnements partagés. ???autoriser plusieurs clients à partager un abonnement à un sujet, un seul client recevant des messages publiés sur ce sujet en utilisant une distribution aléatoire. Pour simuler plusieurs clients MQTT (dans cet exemple, deux clients MQTT) partageant le même abonnement, vous devez ouvrir le client AWS IoT MQTT dans la AWS IoT console à partir de plusieurs navigateurs Web. L'exemple utilisé dans cette section ne concerne pas les exemples utilisés dans Commencer à utiliser les AWS IoT Core didacticiels. Pour plus d'informations, consultez Abonnements partagés.

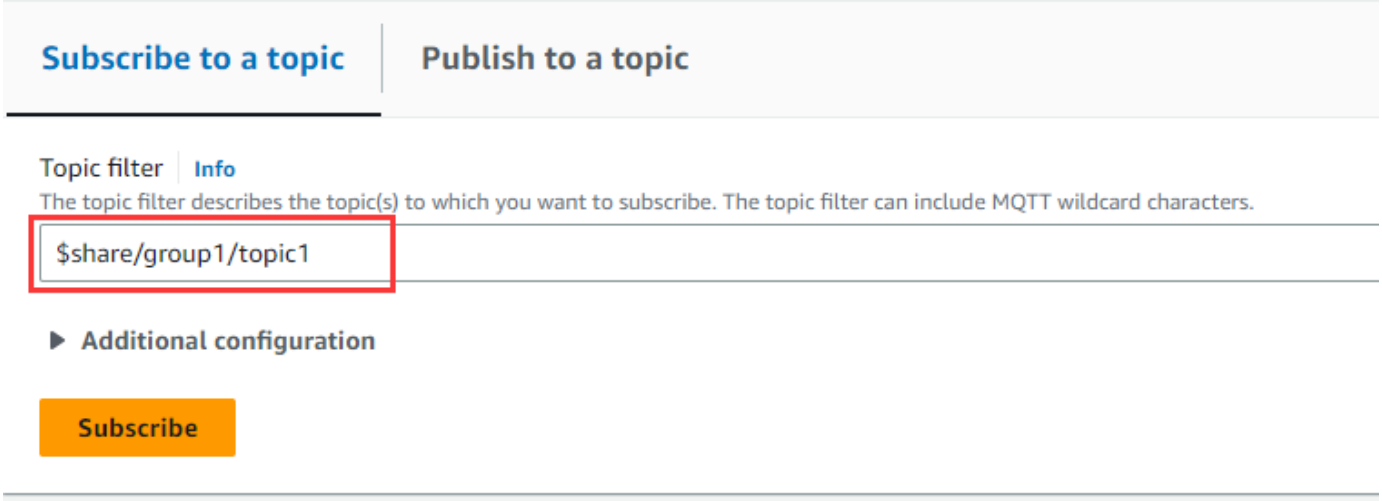
Pour partager un abonnement sur une rubrique MQTT

1. Dans la AWS IoT console, dans le volet de navigation, choisissez Test puis choisissez Client de test MQTT.

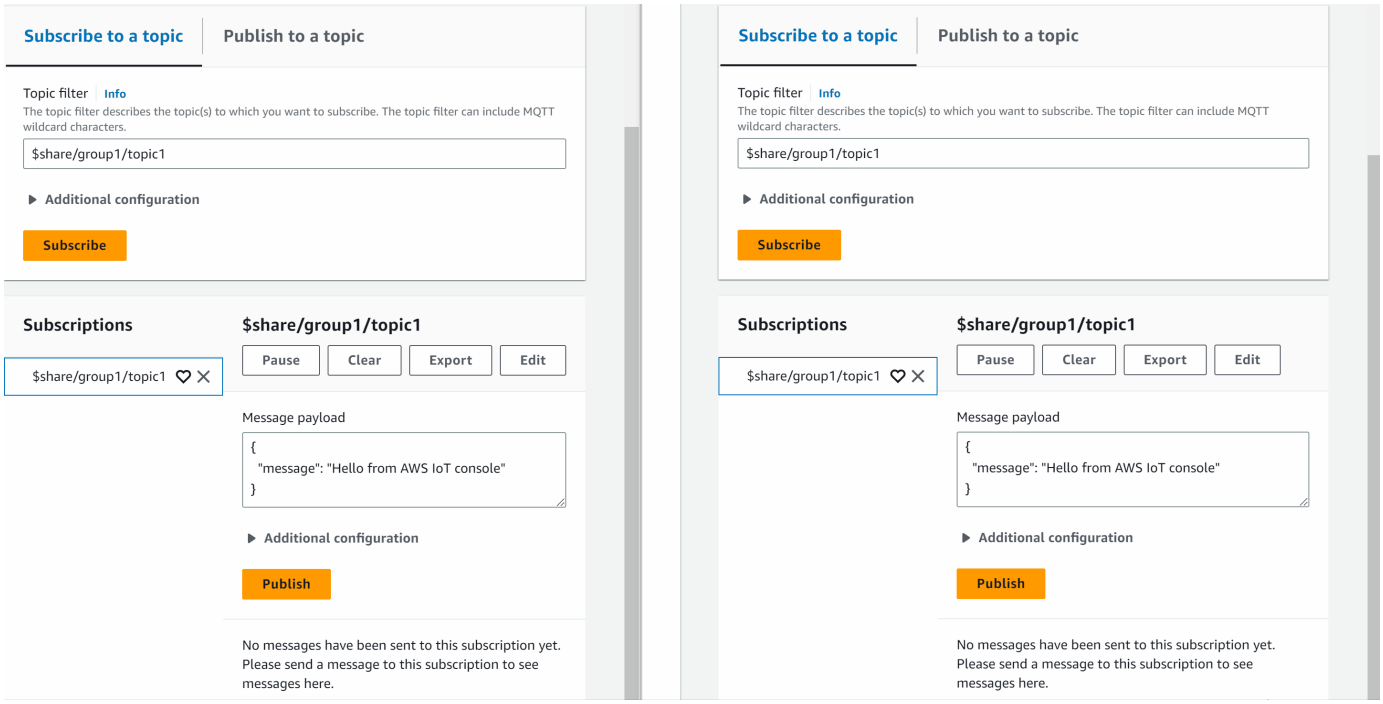
2. Dans l'onglet S'abonner à un sujet, entrez le *topicName* pour vous abonner au sujet sur lequel votre appareil publie. Pour utiliser les abonnements partagés, abonnez-vous au filtre de rubriques d'un abonnement partagé comme suit :

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

Un exemple de filtre de rubrique peut être **$share/group1/topic1**, qui s'abonne à la rubrique **topic1** du message.



3. Ouvrez un autre navigateur Web et répétez les étapes 1 et 2. De cette façon, vous simulez deux clients MQTT différents qui partagent le même abonnement **$share/group1/topic1**.
4. Choisissez un client MQTT, dans l'onglet Publier dans un sujet, dans le champ Nom du sujet, saisissez le contenu *topicName* de votre message. Pour cet exemple, utilisez **topic1**. Essayez de publier le message plusieurs fois. Dans la liste des abonnements des deux clients MQTT, vous devriez être en mesure de voir que les clients reçoivent le message selon une distribution aléatoire. Dans cet exemple, nous publions trois fois le même message « Bonjour depuis AWS IoT la console ». Le client MQTT de gauche a reçu le message deux fois et le client MQTT de droite l'a reçu une fois.


Subscribe to a topic
Publish to a topic
Topic filter Info
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.
$share/group1/topic1
Additional configuration
Subscribe
Subscriptions
$share/group1/topic1
$share/group1/topic1
$share/group1/topic1
Pause
Clear
Export
Edit
Message payload
{
"message": "Hello from AWS IoT console"
}
Additional configuration
Publish
No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

AWS IoT tutoriels

Les AWS IoT didacticiels sont divisés en deux parcours d'apprentissage pour atteindre deux objectifs différents. Choisissez le parcours d’apprentissage le mieux adapté à votre objectif.

- Vous souhaitez créer une idée de AWS IoT solution proof-of-concept pour tester ou démontrer une idée

  Pour présenter les tâches et applications IoT courantes utilisant le AWS IoT Device Client sur vos appareils, suivez le parcours the section called “Création de démos avec le AWS IoT Device Client” d'apprentissage. Le AWS IoT Device Client fournit un logiciel de périphérique avec lequel vous pouvez utiliser vos propres ressources cloud pour démontrer une end-to-end solution avec un minimum de développement.

  Pour plus d'informations sur le client de AWS IoT périphérique, consultez le client de AWS IoT périphérique.
- Vous souhaitez apprendre à créer un logiciel de production pour déployer votre solution

  Pour créer votre propre solution logicielle répondant à vos besoins spécifiques à l'aide d'un SDK pour AWS IoT appareils, suivez le parcours de the section called “Création de solutions à l'aide de l' AWS IoT appareil SDKs” formation.

  Pour plus d'informations sur le AWS IoT périphérique disponible SDKs, consultez???. Pour plus d'informations à ce sujet AWS SDKs, consultez la section Outils sur lesquels vous pouvez vous appuyer AWS.

AWS IoT options de parcours d'apprentissage du didacticiel

- Création de démos avec le AWS IoT Device Client
- Création de solutions à l'aide de l' AWS IoT appareil SDKs

Création de démos avec le AWS IoT Device Client

Les didacticiels de ce parcours d'apprentissage vous expliquent les étapes à suivre pour développer un logiciel de démonstration à l'aide du AWS IoT Device Client. Le AWS IoT Device Client fournit un logiciel qui s'exécute sur votre appareil IoT pour tester et démontrer certains aspects d'une solution IoT basée sur celle-ci AWS IoT.

L'objectif de ces didacticiels est de faciliter l'exploration et l'expérimentation afin que vous puissiez être sûr que votre solution AWS IoT prend en charge votre solution avant de développer le logiciel de votre appareil.

Ce que vous allez apprendre dans ces didacticiels :

- Comment préparer un Raspberry Pi pour une utilisation en tant qu'appareil IoT avec AWS IoT
- Comment démontrer les AWS IoT fonctionnalités à l'aide du client de l' AWS IoT appareil sur votre appareil

Dans ce parcours d'apprentissage, vous allez installer le AWS IoT Device Client sur votre propre Raspberry Pi et créer les AWS IoT ressources dans le cloud pour présenter des idées de solutions IoT. Bien que les didacticiels de ce parcours d'apprentissage présentent les fonctionnalités d'un Raspberry Pi, ils expliquent les objectifs et les procédures pour vous aider à les adapter à d'autres appareils.

Conditions préalables à la création de démos avec le Device Client AWS IoT

Cette section décrit ce dont vous aurez besoin avant de commencer les didacticiels de ce parcours d'apprentissage.

Pour suivre les didacticiels de ce parcours de formation, vous avez besoin des éléments suivants :

- Un Compte AWS

  Vous pouvez utiliser le vôtre existant Compte AWS, si vous en avez un, mais vous devrez peut-être ajouter des rôles ou des autorisations supplémentaires pour utiliser les AWS IoT fonctionnalités utilisées dans ces didacticiels.

  Si vous devez en créer un nouveau Compte AWS, consultezthe section called "Configurez Compte AWS".
- Un Raspberry Pi ou un appareil IoT compatible

  Les didacticiels utilisent un Raspberry Pi car il se présente sous différents formats, il est omniprésent et c'est un appareil de démonstration relativement peu coûteux. Les didacticiels ont été testés sur le Raspberry Pi 3 modèle B+, le Raspberry Pi 4 modèle B et sur une EC2 instance Amazon exécutant Ubuntu Server 20.04 LTS (HVM). Pour utiliser AWS CLI et exécuter les commandes, nous vous recommandons d'utiliser la dernière version du système d'exploitation

Raspberry Pi (Raspberry Pi OS (64 bits) ou OS Lite). Les versions antérieures du système d'exploitation peuvent fonctionner, mais nous ne les avons pas testées.

> **Note**
>
> Les didacticiels expliquent les objectifs de chaque étape afin de vous aider à les adapter au matériel IoT sur lequel nous ne les avons pas essayés. Toutefois, ils ne décrivent pas spécifiquement comment les adapter à d'autres appareils.

- Connaissance du système d'exploitation de l'appareil IoT

  Les étapes de ces didacticiels supposent que vous êtes familiarisé avec l'utilisation des commandes et opérations Linux de base à partir de l'interface de ligne de commande prise en charge par un Raspberry Pi. Si ces opérations ne vous sont pas familières, vous souhaiterez peut-être vous accorder plus de temps pour suivre les didacticiels.

  Pour suivre ces didacticiels, vous devez déjà savoir comment :

  - Effectuez en toute sécurité les opérations de base de l'appareil, telles que l'assemblage et le raccordement des composants, le raccordement de l'appareil aux sources d'alimentation requises, ainsi que l'installation et le retrait de cartes mémoire.
  - Chargez et téléchargez le logiciel système et les fichiers sur l'appareil. Si votre appareil n'utilise pas d'appareil de stockage amovible, tel qu'une carte microSD, vous devez savoir comment vous connecter à votre appareil et comment charger et télécharger le logiciel système et les fichiers sur l'appareil.
  - Connectez votre appareil aux réseaux sur lesquels vous comptez l'utiliser.
  - Connectez-vous à votre appareil depuis un autre ordinateur à l'aide d'un terminal SSH ou d'un programme similaire.
  - Utilisez une interface de ligne de commande pour créer, copier, déplacer, renommer et définir les autorisations des fichiers et des répertoires sur l'appareil.
  - Installez de nouveaux programmes sur l'appareil.
  - Transférez des fichiers depuis et vers votre appareil à l'aide d'outils tels que FTP ou SCP.
- Un environnement de développement et de test pour votre solution IoT

  Les didacticiels décrivent le logiciel et le matériel requis ; ils supposent toutefois que vous serez en mesure d'effectuer des opérations qui ne sont peut-être pas décrites explicitement. Voici des exemples de ce type de matériel et d'opérations :

- Un ordinateur hôte local sur lequel télécharger et stocker des fichiers

  Pour le Raspberry Pi, il s'agit généralement d'un ordinateur personnel ou portable capable de lire et d'écrire sur des cartes mémoire microSD. L'ordinateur hôte local doit :

  - Soyez connecté à Internet.
  - Installz et configurez le AWS CLI.
  - Utilisez un navigateur Web compatible avec la AWS console.
- Un moyen de connecter votre ordinateur hôte local à votre appareil pour communiquer avec lui, entrer des commandes et transférer des fichiers

  Sur le Raspberry Pi, cela se fait souvent en utilisant SSH et SCP depuis l'ordinateur hôte local.
- Un écran et un clavier pour se connecter à l'appareil IoT

  Ils peuvent être utiles, mais ne sont pas obligatoires pour terminer les didacticiels.
- Un moyen pour votre ordinateur hôte local et vos appareils IoT de se connecter à Internet

  Il peut s'agir d'une connexion réseau câblée ou sans fil à un routeur ou à une passerelle connecté à Internet. L'hôte local doit également pouvoir se connecter au Raspberry Pi. Cela peut nécessiter qu'ils soient connectés au même réseau local. Les didacticiels ne peuvent pas vous montrer comment configurer cela pour votre appareil ou la configuration de votre appareil en particulier, mais ils montrent comment tester cette connectivité.
- Accès au routeur de votre réseau local pour voir les appareils connectés

  Pour suivre les didacticiels de ce parcours d'apprentissage, vous devez être en mesure de trouver l'adresse IP de votre appareil IoT.

  Sur un réseau local, cela peut être fait en accédant à l'interface d'administration du routeur réseau auquel vos appareils se connectent. Si vous pouvez attribuer une adresse IP fixe à votre appareil dans le routeur, vous pouvez simplifier la reconnexion après chaque redémarrage de l'appareil.

  Si un clavier et un moniteur sont connectés à l'appareil, ifconfig peut afficher l'adresse IP de l'appareil.

  Si aucune de ces options n'est envisageable, vous devrez trouver un moyen d'identifier l'adresse IP de l'appareil après chaque redémarrage.

Une fois que vous avez tout votre matériel, continuez vers the section called “Préparation à l'utilisation du client IoT Device”.

Tutoriels inclus dans ce parcours d’apprentissage

- Tutoriel : Préparation de vos appareils pour le AWS IoT Device Client
- Didacticiels : Installation et configuration du AWS IoT Device Client
- Tutoriel : Démonstration de la communication des messages MQTT avec AWS IoT Device Client
- Didacticiel : Démonstration d'actions à distance (jobs) avec AWS IoT Device Client
- Tutoriel : Nettoyage après l'exécution des didacticiels AWS IoT Device Client

Tutoriel : Préparation de vos appareils pour le AWS IoT Device Client

Ce didacticiel vous guide tout au long de l'initialisation de votre Raspberry Pi afin de le préparer aux didacticiels suivants de ce parcours d'apprentissage.

L’objectif de ce tutoriel est d’installer la version actuelle du système d’exploitation de l’appareil et de vous assurer que vous pouvez communiquer avec votre appareil dans le contexte de votre environnement de développement.

Prérequis

Avant de commencer ce didacticiel, assurez-vous que les éléments répertoriés the section called “Conditions préalables à la création de démos avec le Device Client AWS IoT” sont disponibles et prêts à être utilisés.

Ce didacticiel vous prendra environ 90 minutes.

Dans ce tutoriel, vous allez :

- Installez et mettez à jour le système d'exploitation de votre appareil.
- Installez et vérifiez tout logiciel supplémentaire nécessaire pour exécuter les didacticiels.
- Testez la connectivité de votre appareil et installez les certificats requis.

Une fois ce didacticiel terminé, le didacticiel suivant prépare votre appareil pour les démos utilisant le AWS IoT Device Client.

Procédures dans ce didacticiel

- Installation et mise à jour du système d'exploitation de l'appareil
- Installez et vérifiez le logiciel requis sur votre appareil
- Testez votre appareil et enregistrez le certificat Amazon CA

Installation et mise à jour du système d'exploitation de l'appareil

Les procédures de cette section expliquent comment initialiser la carte microSD que le Raspberry Pi utilise pour son lecteur système. La carte microSD du Raspberry Pi contient le logiciel de son système d'exploitation (OS) ainsi que de l'espace pour le stockage de ses fichiers d'application. Si vous n'utilisez pas de Raspberry Pi, suivez les instructions de l'appareil pour installer et mettre à jour le logiciel du système d'exploitation de l'appareil.

Après avoir terminé cette section, vous devriez être en mesure de démarrer votre appareil IoT et de vous y connecter depuis le programme du terminal sur votre ordinateur hôte local.

Matériel requis :

- Votre environnement local de développement et de test
- Un Raspberry Pi qui, ou votre appareil IoT, peut se connecter à Internet
- Une carte mémoire microSD d'une capacité d'au moins 8 Go ou d'une capacité de stockage suffisante pour le système d'exploitation et les logiciels requis.

> **Note**
>
> Lorsque vous sélectionnez une carte microSD pour ces exercices, choisissez-en une aussi grande que nécessaire mais aussi petite que possible.
> Une petite carte SD sera plus rapide à sauvegarder et à mettre à jour. Sur le Raspberry Pi, vous n'aurez pas besoin de plus d'une carte microSD de 8 Go pour ces didacticiels. Si vous avez besoin de plus d'espace pour votre application spécifique, les fichiers image plus petits que vous enregistrez dans ces didacticiels peuvent redimensionner le système de fichiers sur une carte plus grande afin d'utiliser tout l'espace pris en charge par la carte de votre choix.

Équipement optionnel :

- Un USB clavier connecté au Raspberry Pi

- Un HDMI moniteur et un câble pour connecter le moniteur au Raspberry Pi

Procédures décrites dans cette section :

- Chargez le système d'exploitation de l'appareil sur la carte microSD
- Démarrez votre appareil IoT avec le nouveau système d'exploitation
- Connectez votre ordinateur hôte local à votre appareil

Chargez le système d'exploitation de l'appareil sur la carte microSD

Cette procédure utilise l'ordinateur hôte local pour charger le système d'exploitation de l'appareil sur une carte microSD.

> **Note**
>
> Si votre appareil n'utilise pas de support de stockage amovible pour son système d'exploitation, installez le système d'exploitation en suivant la procédure correspondant à cet appareil et continuez vers the section called “Démarrez votre appareil IoT”.

Pour installer le système d'exploitation sur votre Raspberry Pi

1. Sur votre ordinateur hôte local, téléchargez et décompressez l'image du système d'exploitation Raspberry Pi que vous souhaitez utiliser. Les dernières versions sont disponibles auprès des systèmes d' https://www.raspberrypi.com/software/exploitation/

   Choisir une version du système d'exploitation Raspberry Pi

   Ce didacticiel utilise la version Raspberry Pi OS Lite, car c'est la plus petite version qui prend en charge les didacticiels de ce parcours d'apprentissage. Cette version du système d'exploitation Raspberry Pi possède uniquement une interface de ligne de commande et n'a pas d'interface utilisateur graphique. Une version du dernier système d'exploitation Raspberry Pi avec une interface utilisateur graphique fonctionnera également avec ces didacticiels ; cependant, les procédures décrites dans ce parcours d'apprentissage utilisent uniquement l'interface de ligne de commande pour effectuer des opérations sur le Raspberry Pi.
2. Insérez votre carte microSD dans l'ordinateur hôte local.
3. À l’aide d’un outil d’imagerie de carte SD, écrivez le fichier image du système d’exploitation décompressé sur la carte microSD.

4. Après avoir écrit l'image du système d'exploitation Raspberry Pi sur la carte microSD :

   a. Ouvrez la BOOT partition de la carte microSD dans une fenêtre de ligne de commande ou dans une fenêtre d'explorateur de fichiers.

   b. Dans la BOOT partition de la carte microSD, dans le répertoire racine, créez un fichier vide nommé `ssh` sans extension ni contenu. Cela indique au Raspberry Pi d'activer les SSH communications au premier démarrage.

5. Éjectez la carte microSD et retirez-la en toute sécurité de l'ordinateur hôte local.

Votre carte microSD est prête à the section called “Démarrez votre appareil IoT”.

Démarrez votre appareil IoT avec le nouveau système d'exploitation

Cette procédure permet d'installer la carte microSD et de démarrer votre Raspberry Pi pour la première fois à l'aide du système d'exploitation téléchargé.

Pour démarrer votre appareil IoT avec le nouveau système d'exploitation

1. L'alimentation étant débranchée de l'appareil, insérez la carte microSD de l'étape précédente, the section called “Chargez le système d'exploitation”, dans le Raspberry Pi.
2. Connectez l'appareil à un réseau filaire.
3. Ces didacticiels interagiront avec votre Raspberry Pi depuis votre ordinateur hôte local à l'aide d'un SSH terminal.

   Si vous souhaitez également interagir directement avec l'appareil, vous pouvez :

   a. Branchez-y un HDMI moniteur pour regarder les messages de la console du Raspberry Pi avant de pouvoir connecter la fenêtre du terminal de votre ordinateur hôte local à votre Raspberry Pi.

   b. USBConnectez-y un clavier si vous souhaitez interagir directement avec le Raspberry Pi.

4. Connectez l'alimentation au Raspberry Pi et attendez environ une minute pour qu'il s'initialise.

   Si vous avez un moniteur connecté à votre Raspberry Pi, vous pouvez suivre le processus de démarrage sur celui-ci.

5. Découvrez l'adresse IP de votre appareil :

   - Si vous avez connecté un HDMI moniteur au Raspberry Pi, l'adresse IP apparaît dans les messages affichés sur le moniteur

- Si vous avez accès au routeur auquel votre Raspberry Pi est connecté, vous pouvez voir son adresse dans l'interface d'administration du routeur.

Une fois que vous avez obtenu l'adresse IP de votre Raspberry Pi, vous êtes prêt à le faire the section called “Connectez votre ordinateur hôte”.

Connectez votre ordinateur hôte local à votre appareil

Cette procédure utilise le programme de terminal sur votre ordinateur hôte local pour vous connecter à votre Raspberry Pi et modifier son mot de passe par défaut.

Pour connecter votre ordinateur hôte local à votre appareil

1. Sur votre ordinateur hôte local, ouvrez le programme du SSH terminal :

   - Windows: `PuTTY`
   - Linux/macOS : `Terminal`

   > **Note**
   >
   > Pu TTY n'est pas installé automatiquement sous Windows. S'il ne se trouve pas sur votre ordinateur, vous devrez peut-être le télécharger et l'installer.

2. Connectez le programme du terminal à l'adresse IP de votre Raspberry Pi et connectez-vous à l'aide de ses informations d'identification par défaut.

   ```
   username: pi
   password: raspberry
   ```

3. Après vous être connecté à votre Raspberry Pi, modifiez le mot de passe pour l'utilisateur `pi`.

   ```
   passwd
   ```

   Suivez les instructions pour modifier le mot de passe.

   ```
   Changing password for pi.
   Current password: raspberry
   New password: YourNewPassword
   Retype new password: YourNewPassword
   ```

```
passwd: password updated successfully
```

Après avoir affiché l'invite de ligne de commande du Raspberry Pi dans la fenêtre du terminal et modifié le mot de passe, vous êtes prêt à continuer the section called “Installation et vérification du logiciel requis”.

Installez et vérifiez le logiciel requis sur votre appareil

Les procédures décrites dans cette section reprennent celles de la section précédente pour mettre à jour le système d'exploitation de votre Raspberry Pi et installer le logiciel sur le Raspberry Pi qui sera utilisé dans la section suivante pour créer et installer le client de AWS IoT périphérique.

Une fois cette section terminée, votre Raspberry Pi disposera up-to-date d'un système d'exploitation, du logiciel requis par les didacticiels de ce parcours d'apprentissage, et il sera configuré en fonction de votre emplacement.

Matériel requis :

- Votre environnement de développement et de test local décrit dans la section précédente
- Le Raspberry Pi que vous avez utilisé dans la section précédente
- La carte mémoire microSD de la section précédente

> **Note**
>
> Le Raspberry Pi Model 3+ et le Raspberry Pi Model 4 peuvent exécuter toutes les commandes décrites dans ce parcours d'apprentissage. Si votre appareil IoT ne parvient pas à compiler le logiciel ou à l'exécuter AWS Command Line Interface, vous devrez peut-être installer les compilateurs requis sur votre ordinateur hôte local pour créer le logiciel, puis le transférer sur votre appareil IoT. Pour obtenir plus d'informations sur l'installation et le développement de logiciels pour votre appareil, consultez la documentation du logiciel de votre appareil.

Procédures décrites dans cette section :

- Mettre à jour le logiciel du système d'exploitation
- Installer les applications et les bibliothèques obligatoires

- (Facultatif) Enregistrez l'image de la carte microSD

Mettre à jour le logiciel du système d'exploitation

Cette procédure met à jour le logiciel du système d'exploitation.

Pour mettre à jour le logiciel du système d'exploitation sur le Raspberry Pi

Effectuez ces étapes dans la fenêtre du terminal de votre ordinateur hôte local.

1. Entrez ces commandes pour mettre à jour le logiciel système de votre Raspberry Pi.

   ```
   sudo apt-get -y update
   sudo apt-get -y upgrade
   sudo apt-get -y autoremove
   ```

2. Mettez à jour les paramètres régionaux et de fuseau horaire du Raspberry Pi (facultatif).

   Entrez cette commande pour mettre à jour les paramètres régionaux et de fuseau horaire de l'appareil.

   ```
   sudo raspi-config
   ```

   a. Pour définir les paramètres régionaux de l'appareil :

      i. Dans Ecran de Outil de configuration logicielle Raspberry Pi (raspi-config), choisissez l'option 5.

         5 Localisation Options Configure language and regional settings

         Utilisez la Tab touche pour passer à, <Select>puis appuyez sur lespace bar.

      ii. Dans le menu des options de localisation, choisissez l'option L1.

         L1 Locale Configure language and regional settings

         Utilisez la touche Tab pour passer à <Select>, puis appuyez sur lespace bar.

      iii. Dans la liste des options de paramètres régionaux, choisissez les paramètres régionaux que vous souhaitez installer sur votre Raspberry Pi en utilisant les touches fléchées pour faire défiler la page et space bar en sélectionnant celles que vous souhaitez.

Aux États-Unis, **en_US.UTF-8** est un bon choix.

iv. Après avoir sélectionné les paramètres régionaux pour votre appareil, utilisez la touche Tab pour choisir<OK>, puis appuyez sur le space bar pour afficher la page de confirmation de configuration des paramètres régionaux.

b. Pour définir le fuseau horaire de l'appareil, procédez comme suit :

i. Dans l'écran raspi-config, choisissez l'option 5.

5 Localisation Options Configure language and regional settings

Utilisez la touche Tab pour passer à <Select>, puis appuyez sur lespace bar.

ii. Dans le menu des options de localisation, utilisez la touche fléchée pour sélectionner l'option L2 :

L2 time zone Configure time zone

Utilisez la touche Tab pour passer à <Select>, puis appuyez sur lespace bar.

iii. Dans le menu Configuration de tzdata, choisissez votre zone géographique dans la liste.

Utilisez la touche Tab pour passer à <OK>, puis appuyez sur space bar.

iv. Dans la liste des villes, utilisez les flèches pour sélectionner une ville dans votre fuseau horaire.

Pour définir le fuseau horaire, utilisez la touche Tab pour passer à <OK>, puis appuyez sur space bar.

c. Lorsque vous avez terminé de mettre à jour les paramètres, utilisez la touche Tab pour accéder à<Finish>, puis appuyez sur la touche space bar pour fermer l'application raspi-config.

3. Entrez cette commande pour redémarrer votre Raspberry Pi.

```
sudo shutdown -r 0
```

4. Attendez que votre Raspberry Pi redémarre.

5. Après le redémarrage de votre Raspberry Pi, reconnectez la fenêtre du terminal de votre ordinateur hôte local à votre Raspberry Pi.

Le logiciel de votre système Raspberry Pi est maintenant configuré et vous êtes prêt à continuer the section called “Installation d'applications et de bibliothèques”.

Installer les applications et les bibliothèques obligatoires

Cette procédure installe le logiciel d'application et les bibliothèques utilisés dans les didacticiels suivants.

Si vous utilisez un Raspberry Pi ou si vous pouvez compiler le logiciel requis sur votre appareil IoT, effectuez ces étapes dans la fenêtre du terminal de votre ordinateur hôte local. Si vous devez compiler le logiciel pour votre appareil IoT sur votre ordinateur hôte local, consultez la documentation du logiciel de votre appareil IoT pour savoir comment effectuer ces étapes sur votre appareil.

Pour installer le logiciel d'application et les bibliothèques sur votre Raspberry Pi

1. Entrez cette commande pour installer le logiciel d'application et les bibliothèques.

   ```
   sudo apt-get -y install build-essential libssl-dev cmake unzip git python3-pip
   ```

2. Entrez ces commandes pour vérifier que la bonne version du logiciel a été installée.

   ```
   gcc --version
   cmake --version
   openssl version
   git --version
   ```

3. Vérifiez que les versions suivantes du logiciel d'application sont installées :

   - `gcc`: 9.3.0 ou version ultérieure
   - `cmake`: 3.10.x ou version ultérieure
   - `OpenSSL`: 1.1.1 ou version ultérieure
   - `git`: 2.20.1 ou version ultérieure

Si votre Raspberry Pi possède des versions acceptables du logiciel d'application requis, vous êtes prêt à continuer the section called “(Facultatif) Enregistrer l'image microSD”.

(Facultatif) Enregistrez l'image de la carte microSD

Tout au long des didacticiels de ce parcours d'apprentissage, vous rencontrerez ces procédures pour enregistrer une copie de l'image de la carte microSD du Raspberry Pi dans un fichier sur votre

ordinateur hôte local. Bien qu'elles soient encouragées, ce ne sont pas des tâches obligatoires. En enregistrant l'image de la carte microSD à l'endroit suggéré, vous pouvez ignorer les procédures qui précèdent le point de sauvegarde dans ce parcours d'apprentissage, ce qui peut vous faire gagner du temps si vous ressentez le besoin de réessayer quelque chose. La conséquence de ne pas enregistrer périodiquement l'image de la carte microSD est que vous devrez peut-être redémarrer les didacticiels du parcours d'apprentissage depuis le début si votre carte microSD est endommagée ou si vous configurez accidentellement une application ou ses paramètres de manière incorrecte.

À ce stade, la carte microSD de votre Raspberry Pi dispose d'un système d'exploitation mis à jour et du logiciel d'application de base chargé. Vous pouvez gagner le temps qu'il vous a fallu pour effectuer les étapes précédentes en enregistrant maintenant le contenu de la carte microSD dans un fichier. Avoir l'image actuelle de l'image de la carte microSD de votre appareil vous permet de partir de ce point pour continuer ou réessayer un didacticiel ou une procédure sans avoir besoin d'installer et de mettre à jour le logiciel à partir de zéro.

Pour enregistrer l'image de la carte microSD dans un fichier

1. Entrez cette commande pour arrêter le Raspberry Pi.

   ```
   sudo shutdown -h 0
   ```

2. Une fois le Raspberry Pi complètement éteint, coupez son alimentation.
3. Retirez la carte microSD du Raspberry Pi.
4. Sur votre ordinateur hôte local :

   a. Insérez la carte microSD.

   b. À l’aide de votre outil d’imagerie de carte SD, enregistrez l’image de la carte microSD dans un fichier.

   c. Une fois l'image de la carte microSD enregistrée, éjectez-la de l'ordinateur hôte local.
5. L'alimentation du Raspberry Pi étant débranchée, insérez la carte microSD dans le Raspberry Pi.
6. Alimentez le Raspberry Pi.
7. Après avoir attendu environ une minute, sur l'ordinateur hôte local, reconnectez la fenêtre du terminal sur votre ordinateur hôte local qui était connecté à votre Raspberry Pi., puis connectez-vous au Raspberry Pi.

Testez votre appareil et enregistrez le certificat Amazon CA

Les procédures décrites dans cette section reprennent celles de la section précédente pour installer le certificat AWS Command Line Interface et le certificat de l'autorité de certification utilisés pour authentifier vos connexions. AWS IoT Core

Après avoir terminé cette section, vous saurez que votre Raspberry Pi dispose du logiciel système nécessaire pour installer le client du AWS IoT périphérique et qu'il dispose d'une connexion Internet fonctionnelle.

Matériel requis :

- Votre environnement de développement et de test local décrit dans la section précédente
- Le Raspberry Pi que vous avez utilisé dans la section précédente
- La carte mémoire microSD de la section précédente

Procédures décrites dans cette section :

- Installez le AWS Command Line Interface
- Configurez vos Compte AWS informations d'identification
- Téléchargez le certificat Amazon Root CA
- (Facultatif) Enregistrez l'image de la carte microSD

Installez le AWS Command Line Interface

Cette procédure permet de l'installer AWS CLI sur votre Raspberry Pi.

Si vous utilisez un Raspberry Pi ou si vous pouvez compiler un logiciel sur votre appareil IoT, effectuez ces étapes dans la fenêtre du terminal sur votre ordinateur hôte local. Si vous devez compiler un logiciel pour votre appareil IoT sur votre ordinateur hôte local, consultez la documentation logicielle de votre appareil IoT pour obtenir des informations sur les bibliothèques dont il a besoin.

Pour l'installer AWS CLI sur votre Raspberry Pi

1. Exécutez ces commandes pour télécharger et installer le AWS CLI.

```
export PATH=$PATH:~/.local/bin # configures the path to include the directory with
 the AWS CLI
```

```
git clone https://github.com/aws/aws-cli.git # download the AWS CLI code from
 GitHub
cd aws-cli && git checkout v2 # go to the directory with the repo and checkout
 version 2
pip3 install -r requirements.txt # install the prerequisite software
```

2. Exécutez cette commande pour installer le AWS CLI. Cette commande peut prendre jusqu'à 15 minutes.

   ```
   pip3 install . # install the AWS CLI
   ```

3. Exécutez cette commande pour vérifier que la bonne version de AWS CLI a été installée.

   ```
   aws --version
   ```

   La version de AWS CLI doit être 2.2 ou ultérieure.

Si la version actuelle est AWS CLI affichée, vous êtes prêt à continuerthe section called “Configurer les informations d'identification du compte”.

Configurez vos Compte AWS informations d'identification

Dans cette procédure, vous allez obtenir des Compte AWS informations d'identification et les ajouter pour les utiliser sur votre Raspberry Pi.

Pour ajouter vos Compte AWS informations d'identification à votre appareil

1. Obtenez un identifiant de clé d'accès et une clé d'accès secrète auprès de vous Compte AWS pour les authentifier AWS CLI sur votre appareil.

   Si vous êtes nouveau dans ce domaine AWS IAM, le https://aws.amazon.com/premiumsupport/centre de connaissances/create-access-key/décrit le processus à exécuter dans la AWS console pour créer des AWS IAM informations d'identification à utiliser sur votre appareil.

2. Dans la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi et avec les informations d'identification de la clé d'accès et de la clé d'accès secrète de votre appareil :

   a. Exécutez l'application de AWS configuration à l'aide de cette commande :

      ```
      aws configure
      ```

b. Entrez vos informations d'identification et de configuration lorsque vous y êtes invité :

```
AWS Access Key ID: your Access Key ID
AWS Secret Access Key: your Secret Access Key
Default region name: your Région AWS code
Default output format: json
```

3. Exécutez cette commande pour tester l'accès de votre appareil à votre AWS IoT Core terminal Compte AWS et à votre terminal.

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

Il doit renvoyer votre point de terminaison de AWS IoT données Compte AWS spécifique, comme dans cet exemple :

```
{
    "endpointAddress": "a3EXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

Si vous voyez votre point de terminaison de AWS IoT données Compte AWS spécifique, votre Raspberry Pi dispose de la connectivité et des autorisations nécessaires pour continuerthe section called “Téléchargez le certificat Amazon Root CA”.

> **Important**
>
> Vos Compte AWS informations d'identification sont désormais stockées sur la carte microSD de votre Raspberry Pi. Bien que cela facilite les interactions futures avec AWS vous et les logiciels que vous allez créer dans ces didacticiels, elles seront également enregistrées et dupliquées par défaut dans toutes les images de carte microSD que vous créerez après cette étape.
>
> Pour garantir la sécurité de vos Compte AWS informations d'identification, avant d'enregistrer d'autres images de carte microSD, pensez à effacer les informations d'identification en exécutant à `aws configure` nouveau et en saisissant des caractères aléatoires pour l'ID de clé d'accès et la clé d'accès secrète afin d'éviter que vos Compte AWS informations d'identification ne soient compromises.
>
> Si vous constatez que vous avez enregistré vos Compte AWS informations d'identification par inadvertance, vous pouvez les désactiver dans la console. AWS IAM

Téléchargez le certificat Amazon Root CA

Cette procédure télécharge et enregistre une copie d'un certificat de l'Amazon Root Certificate Authority (CA). Le téléchargement de ce certificat est enregistré pour être utilisé dans les didacticiels suivants et permet également de tester la connectivité de votre appareil aux services AWS .

Pour télécharger et enregistrer le certificat Amazon Root CA

1. Exécutez cette commande pour créer un répertoire pour le certificat.

   ```
   mkdir ~/certs
   ```

2. Exécutez cette commande pour télécharger le certificat Amazon Root CA.

   ```
   curl -o ~/certs/AmazonRootCA1.pem https://www.amazontrust.com/repository/
   AmazonRootCA1.pem
   ```

3. Exécutez ces commandes pour définir l'accès au répertoire du certificat et à son fichier.

   ```
   chmod 745 ~
   chmod 700 ~/certs
   chmod 644 ~/certs/AmazonRootCA1.pem
   ```

4. Exécutez cette commande pour voir le fichier de certificat CA dans le nouveau répertoire.

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

   Vous devriez voir une entrée comme celle-ci. La date et l'heure seront différentes ; cependant, la taille du fichier et toutes les autres informations doivent être les mêmes que celles indiquées ici.

   ```
   -rw-r--r-- 1 pi pi 1188 Oct 28 13:02 AmazonRootCA1.pem
   ```

   Si la taille du fichier n’est pas 1188, vérifiez les paramètres de commande curl. Vous avez peut-être téléchargé un fichier incorrect.

(Facultatif) Enregistrez l'image de la carte microSD

À ce stade, la carte microSD de votre Raspberry Pi dispose d'un système d'exploitation mis à jour et du logiciel d'application de base chargé.

Pour enregistrer l'image de la carte microSD dans un fichier

1. Dans la fenêtre du terminal de votre ordinateur hôte local, effacez vos informations d'identification AWS .

   a. Exécutez l'application de AWS configuration à l'aide de cette commande :

   ```
   aws configure
   ```

   b. Remplacez vos informations d'identification lorsque vous y êtes invité. Vous pouvez laisser le nom de la région par défaut et le format de sortie par défaut tels quels en appuyant sur Entrée.

   ```
   AWS Access Key ID [****************YT2H]: XYXYXYXYX
   AWS Secret Access Key [****************9plH]: XYXYXYXYX
   Default region name [us-west-2]:
   Default output format [json]:
   ```

2. Entrez cette commande pour arrêter le Raspberry Pi.

   ```
   sudo shutdown -h 0
   ```

3. Une fois le Raspberry Pi complètement arrêté, retirez son connecteur d'alimentation.
4. Retirez la carte microSD de votre appareil.
5. Sur votre ordinateur hôte local :

   a. Insérez la carte microSD.

   b. À l’aide de votre outil d’imagerie de carte SD, enregistrez l’image de la carte microSD dans un fichier.

   c. Une fois l'image de la carte microSD enregistrée, éjectez-la de l'ordinateur hôte local.

6. L'alimentation du Raspberry Pi étant débranchée, insérez la carte microSD dans le Raspberry Pi.
7. Mettez l'appareil sous tension.
8. Après environ une minute, sur l'ordinateur hôte local, redémarrez la session de fenêtre du terminal et connectez-vous à l'appareil.

   Ne saisissez pas encore vos Compte AWS informations d'identification.

Après avoir redémarré et connecté à votre Raspberry Pi, vous êtes prêt à continuer sur the section called “Installation et configuration d'un client pour appareils IoT”.

Didacticiels : Installation et configuration du AWS IoT Device Client

Ce didacticiel explique l'installation et la configuration du client de AWS IoT périphérique, ainsi que la création des AWS IoT ressources que vous utiliserez dans cette démo et dans d'autres.

Pour démarrer ce didacticiel :

- Préparez votre ordinateur hôte local et le Raspberry Pi du didacticiel précédent.

Ce didacticiel peut prendre jusqu'à 90 minutes.

Lorsque vous avez terminé avec cette rubrique :

- Votre appareil IoT sera prêt à être utilisé dans d'autres démonstrations de AWS IoT Device Client.
- Vous aurez approvisionné votre appareil IoT. AWS IoT Core
- Vous aurez téléchargé et installé le client de l' AWS IoT appareil sur votre appareil.
- Vous aurez enregistré une image de la carte microSD de votre appareil qui pourra être utilisée dans les didacticiels suivants.

Matériel requis :

- Votre environnement de développement et de test local décrit dans la section précédente
- Le Raspberry Pi que vous avez utilisé dans la section précédente
- La carte mémoire microSD du Raspberry Pi que vous avez utilisée dans la section précédente

Procédures dans ce didacticiel

- Téléchargez et enregistrez le client de AWS IoT l'appareil
- Approvisionnez votre Raspberry Pi AWS IoT
- Configurer le client du AWS IoT périphérique pour tester la connectivité

Téléchargez et enregistrez le client de AWS IoT l'appareil

Les procédures décrites dans cette section permettent de télécharger le client de AWS IoT périphérique, de le compiler et de l'installer sur votre Raspberry Pi. Après avoir testé l'installation, vous pouvez enregistrer l'image de la carte microSD du Raspberry Pi pour l'utiliser ultérieurement lorsque vous souhaiterez réessayer les didacticiels.

Procédures décrites dans cette section :

- Téléchargez et créez le AWS IoT Device Client
- Créez les répertoires utilisés par les didacticiels
- (Facultatif) Enregistrez l'image de la carte microSD

Téléchargez et créez le AWS IoT Device Client

Cette procédure installe le AWS IoT Device Client sur votre Raspberry Pi.

Exécutez ces commandes dans la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi.

Pour installer le client de AWS IoT périphérique sur votre Raspberry Pi

1. Entrez ces commandes pour télécharger et créer le client de AWS IoT périphérique sur votre Raspberry Pi.

   ```
   cd ~
   git clone https://github.com/awslabs/aws-iot-device-client aws-iot-device-client
   mkdir ~/aws-iot-device-client/build && cd ~/aws-iot-device-client/build
   cmake ../
   ```

2. Exécutez cette commande pour créer le client de AWS IoT périphérique. Cette commande peut prendre jusqu'à 15 minutes.

   ```
   cmake --build . --target aws-iot-device-client
   ```

   Les messages d'avertissement affichés lors de la compilation du AWS IoT Device Client peuvent être ignorés.

   Ces didacticiels ont été testés avec le AWS IoT Device Client intégrégcc, version (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110 sur la version du 30 octobre 2021 du système d'exploitation

Raspberry Pi (bullseye) activéegcc, version (Raspbian 8.3.0-6+rpi1) 8.3.0 sur la version du 7 mai 2021 du système d'exploitation Raspberry Pi (buster).

3. Une fois que le AWS IoT Device Client a terminé de construire, testez-le en exécutant cette commande.

```
./aws-iot-device-client --help
```

Si vous voyez l'aide en ligne de commande pour le AWS IoT client de AWS IoT périphérique, celui-ci a été créé avec succès et est prêt à être utilisé.

Créez les répertoires utilisés par les didacticiels

Cette procédure crée les répertoires du Raspberry Pi qui seront utilisés pour stocker les fichiers utilisés par les didacticiels de ce parcours d'apprentissage.

Pour créer les répertoires utilisés par les didacticiels de ce parcours d'apprentissage, procédez comme suit :

1. Exécutez ces commandes pour créer les répertoires requis.

```
mkdir ~/dc-configs
mkdir ~/policies
mkdir ~/messages
mkdir ~/certs/testconn
mkdir ~/certs/pubsub
mkdir ~/certs/jobs
```

2. Exécutez ces commandes pour définir les autorisations sur les nouveaux répertoires.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 700 ~/certs/pubsub
chmod 700 ~/certs/jobs
```

Après avoir créé ces annuaires et défini leur autorisation, passez à the section called “(Facultatif) Enregistrez l'image de la carte microSD”.

(Facultatif) Enregistrez l'image de la carte microSD

À ce stade, la carte microSD de votre Raspberry Pi dispose d'un système d'exploitation mis à jour, du logiciel d'application de base et du client de AWS IoT périphérique.

Si vous souhaitez réessayer ces exercices et didacticiels, vous pouvez ignorer les procédures précédentes en écrivant l'image de la carte microSD que vous avez enregistrée avec cette procédure sur une nouvelle carte microSD et en continuant les didacticiels à partir de cette procédure the section called “Provisionnez votre Raspberry Pi”.

Pour enregistrer l'image de la carte microSD dans un fichier :

Dans la fenêtre du terminal sur votre ordinateur hôte local connecté à votre Raspberry Pi :

1. Vérifiez que vos Compte AWS informations d'identification n'ont pas été enregistrées.

   a. Exécutez l'application de AWS configuration à l'aide de cette commande :

   ```
   aws configure
   ```

   b. Si vos informations d'identification ont été enregistrées (si elles sont affichées dans l'invite), entrez la chaîne **XYXYXYXYX** lorsque vous y êtes invité, comme indiqué ici. Laissez le nom de la région par défaut et le format de sortie par défaut vides.

   ```
   AWS Access Key ID [****************YXYX]: XYXYXYXYX
   AWS Secret Access Key [****************YXYX]: XYXYXYXYX
   Default region name:
   Default output format:
   ```

2. Entrez cette commande pour arrêter le Raspberry Pi.

   ```
   sudo shutdown -h 0
   ```

3. Une fois le Raspberry Pi complètement arrêté, retirez son connecteur d'alimentation.
4. Retirez la carte microSD de votre appareil.
5. Sur votre ordinateur hôte local :

   a. Insérez la carte microSD.

   b. À l’aide de votre outil d’imagerie de carte SD, enregistrez l’image de la carte microSD dans un fichier.

c. Une fois l'image de la carte microSD enregistrée, éjectez-la de l'ordinateur hôte local.

Vous pouvez continuer avec cette carte microSD the section called “Provisionnez votre Raspberry Pi”.

Approvisionnez votre Raspberry Pi AWS IoT

Les procédures décrites dans cette section commencent par l'image microSD enregistrée sur laquelle le client de AWS IoT périphérique est installé AWS CLI et créent les AWS IoT ressources et les certificats de périphérique qui approvisionnent votre Raspberry Pi. AWS IoT

Installez la carte microSD dans votre Raspberry Pi

Cette procédure permet d'installer la carte microSD avec le logiciel nécessaire chargé et configuré dans le Raspberry Pi et de la configurer afin que Compte AWS vous puissiez poursuivre les didacticiels de ce parcours d'apprentissage.

Utilisez une carte microSD de the section called “(Facultatif) Enregistrez l'image de la carte microSD” contenant le logiciel nécessaire pour les exercices et tutoriels de ce parcours d'apprentissage.

Pour installer la carte microSD dans votre Raspberry Pi

1. L'alimentation du Raspberry Pi étant débranchée, insérez la carte microSD dans le Raspberry Pi.
2. Alimentez le Raspberry Pi.
3. Après environ une minute, sur l'ordinateur hôte local, redémarrez la session de fenêtre du terminal et connectez-vous au Raspberry Pi.
4. Sur votre ordinateur hôte local, dans la fenêtre du terminal, et avec les informations d'identification de la clé d'accès et de la clé d'accès secrète de votre Raspberry Pi :

   a. Exécutez l'application de AWS configuration à l'aide de cette commande :

   ```
   aws configure
   ```

   b. Entrez vos Compte AWS informations d'identification et de configuration lorsque vous y êtes invité :

   ```
   AWS Access Key ID [****************YXYX]: your Access Key ID
   AWS Secret Access Key [****************YXYX]: your Secret Access Key
   Default region name [us-west-2]: your Région AWS code
   ```

```
Default output format [json]: json
```

Une fois que vous avez restauré vos Compte AWS informations d'identification, vous êtes prêt à continuerthe section called “Approvisionnez votre appareil en AWS IoT Core”.

Approvisionnez votre appareil en AWS IoT Core

Les procédures décrites dans cette section créent les AWS IoT ressources qui approvisionnent votre Raspberry Pi AWS IoT. Au fur et à mesure que vous créez ces ressources, il vous sera demandé d'enregistrer diverses informations. Ces informations sont utilisées par la configuration du client du AWS IoT périphérique dans la procédure suivante.

Pour que votre Raspberry Pi fonctionne AWS IoT, il doit être configuré. Le provisionnement est le processus de création et de configuration des AWS IoT ressources nécessaires pour prendre en charge votre Raspberry Pi en tant qu'appareil IoT.

Une fois votre Raspberry Pi allumé et redémarré, connectez la fenêtre du terminal de votre ordinateur hôte local au Raspberry Pi et effectuez ces procédures.

Procédures décrites dans cette section :

- Création et téléchargement de fichiers de certificat d'appareil
- Créez des AWS IoT ressources

Création et téléchargement de fichiers de certificat d'appareil

Cette procédure crée les fichiers de certificat de l'appareil pour cette démonstration.

Pour créer et télécharger les fichiers de certificat d'appareil pour votre Raspberry Pi

1. Dans la fenêtre du terminal de votre ordinateur hôte local, entrez ces commandes pour créer les fichiers de certificat de périphérique pour votre appareil.

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

La commande renvoie une réponse comme celle-ci. Enregistrez la valeur *certificateArn* pour une utilisation ultérieure.

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

2. Entrez les commandes suivantes pour définir les autorisations sur le répertoire des certificats et ses fichiers.

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 644 ~/certs/testconn/*
chmod 600 ~/certs/testconn/private.pem.key
```

3. Exécutez cette commande pour vérifier les autorisations sur vos répertoires et fichiers de certificats.

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

Le résultat de la commande doit être identique à ce que vous voyez ici, sauf que les dates et heures du fichier seront différentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

À ce stade, les fichiers de certificat de l'appareil sont installés sur votre Raspberry Pi et vous pouvez continuer the section called “Créez des AWS IoT ressources”.

Créez des AWS IoT ressources

Cette procédure approvisionne votre appareil en AWS IoT créant les ressources dont il a besoin pour accéder aux AWS IoT fonctionnalités et aux services.

Pour approvisionner votre appareil dans AWS IoT

1. Dans la fenêtre du terminal de votre ordinateur hôte local, entrez la commande suivante pour obtenir l'adresse du point de terminaison des données de l'appareil pour votre Compte AWS.

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

   La commande des étapes précédentes renvoie une réponse comme celle-ci. Enregistrez la valeur *endpointAddress* pour une utilisation ultérieure.

   ```
   {
       "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Entrez cette commande pour créer une ressource d' AWS IoT objets pour votre Raspberry Pi.

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

   Si votre ressource d' AWS IoT objet a été créée, la commande renvoie une réponse comme celle-ci.

   ```
   {
       "thingName": "DevCliTestThing",
       "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/DevCliTestThing",
       "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
   }
   ```

3. Dans la fenêtre du terminal :

   a. Ouvrez un éditeur de texte comme nano.

   b. Copiez ce document de politique JSON et collez-le dans votre éditeur de texte ouvert.

> **Note**
>
> Ce document de politique accorde généreusement à chaque ressource l'autorisation de se connecter, de recevoir, de publier et de s'abonner. Normalement, les politiques n'autorisent que des ressources spécifiques à effectuer des actions spécifiques. Toutefois, pour le test initial de connectivité de l'appareil, cette politique trop générale et permissive est utilisée pour minimiser les risques de problème d'accès lors de ce test. Dans les didacticiels suivants, des documents de politique plus restreints seront utilisés pour démontrer les meilleures pratiques en matière de conception de politiques.

c. Enregistrez le fichier dans votre éditeur de texte sous le nom **`~/policies/dev_cli_test_thing_policy.json`**

4. Exécutez cette commande pour utiliser le document de stratégie décrit dans les étapes précédentes afin de créer une AWS IoT stratégie.

```
aws iot create-policy \
--policy-name "DevCliTestThingPolicy" \
--policy-document "file://~/policies/dev_cli_test_thing_policy.json"
```

Si la politique est créée, la commande renvoie une réponse comme celle-ci.

```
{
    "policyName": "DevCliTestThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/DevCliTestThingPolicy",
    "policyDocument": "{\n    \"Version\": \"2012-10-17\",\n    \"Statement\": [\n
        {\n            \"Effect\": \"Allow\",\n            \"Action\": [\n
           \"iot:Publish\",\n               \"iot:Subscribe\",\n
 \"iot:Receive\",\n               \"iot:Connect\"\n           ],\n
 \"Resource\": [\n               \"*\"\n            ]\n        }\n    ]\n}\n",
    "policyVersionId": "1"
}
```

5. Exécutez cette commande pour associer la politique au certificat de l'appareil. Remplacez *`certificateArn`* par la valeur `certificateArn` que vous avez enregistrée précédemment.

```
aws iot attach-policy \
--policy-name "DevCliTestThingPolicy" \
```

```
--target "certificateArn"
```

Si elle aboutit, cette commande ne renvoie rien.

6. Exécutez cette commande pour associer le certificat de l'appareil à la ressource d' AWS IoT objet. Remplacez *certificateArn* par la valeur `certificateArn` que vous avez enregistrée précédemment.

```
aws iot attach-thing-principal \
--thing-name "DevCliTestThing" \
--principal "certificateArn"
```

Si elle aboutit, cette commande ne renvoie rien.

Une fois que vous avez correctement configuré votre appareil AWS IoT, vous êtes prêt à the section called “Configurer le client du périphérique et tester la connectivité” continuer.

Configurer le client du AWS IoT périphérique pour tester la connectivité

Les procédures décrites dans cette section configurent le client du AWS IoT périphérique pour publier un message MQTT depuis votre Raspberry Pi.

Procédures décrites dans cette section :

- Créez le fichier de configuration
- Ouvrez le client de test MQTT
- Exécuter AWS IoT le client du périphérique

Créez le fichier de configuration

Cette procédure crée le fichier de configuration pour tester le client du AWS IoT périphérique.

Pour créer le fichier de configuration permettant de tester le client du AWS IoT périphérique

- Dans la fenêtre du terminal sur votre ordinateur hôte local connecté à votre Raspberry Pi :

  a. Entrez ces commandes pour créer un répertoire pour les fichiers de configuration et définir l'autorisation sur le répertoire :

```
mkdir ~/dc-configs
```

```
chmod 745 ~/dc-configs
```

b. Ouvrez un éditeur de texte commenano.

c. Copiez ce document JSON et collez-le dans votre éditeur de texte ouvert.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/testconn/device.pem.crt",
  "key": "~/certs/testconn/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "DevCliTestThing",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": ""
    }
```

```
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

d. Remplacez la *endpoint* valeur par le point de terminaison de données de l'appareil dans Compte AWS lequel vous l'avez trouvéthe section called “Approvisionnez votre appareil en AWS IoT Core”.

e. Enregistrez le fichier dans votre éditeur de texte sous le nom **~/dc-configs/dc-testconn-config.json**.

f. Exécutez cette commande pour définir les autorisations sur le nouveau fichier de configuration.

```
chmod 644 ~/dc-configs/dc-testconn-config.json
```

Après avoir enregistré le fichier, vous êtes prêt à continuer the section called “Ouvrez le client de test MQTT”.

Ouvrez le client de test MQTT

Cette procédure prépare le client de test MQTT dans la AWS IoT console à s'abonner au message MQTT que le client de AWS IoT périphérique publie lors de son exécution.

Pour préparer le client de test MQTT à s'abonner à tous les messages MQTT

1. Sur votre ordinateur hôte local, dans la AWS IoT console, choisissez le client de test MQTT.
2. Dans l'onglet S'abonner à un sujet, dans le filtre de sujet, entrez **#** ((un seul signe dièse), puis choisissez S'abonner pour vous abonner à chaque sujet MQTT.
3. Sous l'étiquette Abonnements, confirmez que vous voyez **#** (un seul signe dièse).

Laissez la fenêtre contenant le client de test MQTT ouverte pendant que vous continuez the section called “Exécuter AWS IoT le client du périphérique”.

Exécuter AWS IoT le client du périphérique

Cette procédure exécute le client de AWS IoT périphérique afin qu'il publie un seul message MQTT que le client de test MQTT reçoit et affiche.

Pour envoyer un message MQTT depuis le client du AWS IoT périphérique

1. Assurez-vous que la fenêtre du terminal connectée à votre Raspberry Pi et la fenêtre avec le client de test MQTT sont visibles pendant que vous effectuez cette procédure.
2. Dans la fenêtre du terminal, entrez ces commandes pour exécuter le client de AWS IoT périphérique à l'aide du fichier de configuration créé dansthe section called “Créez le fichier de configuration”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-testconn-config.json
   ```

   Dans la fenêtre du terminal, le client du AWS IoT périphérique affiche les messages d'information et les éventuelles erreurs survenant lors de son exécution.

   Si aucune erreur n'est affichée dans la fenêtre du terminal, passez en revue le client de test MQTT.
3. Dans le client de test MQTT, dans la fenêtre Abonnements, consultez le Hello World ! message envoyé à la rubrique du message `test/dc/pubtopic`.
4. Si le client de l' AWS IoT appareil n'affiche aucune erreur et que vous voyez Hello World ! envoyé au `test/dc/pubtopic` message dans le client de test MQTT, vous avez démontré une connexion réussie.
5. Dans la fenêtre du terminal, entrez **^C** (Ctrl-C) pour arrêter le AWS IoT Device Client.

Après avoir démontré que le AWS IoT Device Client fonctionne correctement sur votre Raspberry Pi et qu'il peut communiquer avec lui AWS IoT, vous pouvez passer authe section called “Communiquez avec le client Device à l'aide de MQTT”.

Tutoriel : Démonstration de la communication des messages MQTT avec AWS IoT Device Client

Ce didacticiel explique comment le AWS IoT Device Client peut s'abonner à des messages MQTT et les publier, qui sont couramment utilisés dans les solutions IoT.

Pour démarrer ce didacticiel :

- Faites configurer votre ordinateur hôte local et votre Raspberry Pi comme utilisés dans la section précédente.

  Si vous avez enregistré l'image de la carte microSD après avoir installé le client du AWS IoT périphérique, vous pouvez utiliser une carte microSD avec cette image avec votre Raspberry Pi.
- Si vous avez déjà lancé cette démo, vérifiez si vous ??? souhaitez supprimer toutes les AWS IoT ressources que vous avez créées lors des exécutions précédentes afin d'éviter les erreurs liées aux ressources dupliquées.

Ce didacticiel vous prendra environ 45 minutes.

Lorsque vous avez terminé avec cette rubrique :

- Vous aurez démontré les différentes manières dont votre appareil IoT peut s'abonner à des messages MQTT AWS IoT et publier des messages MQTT à destination de ceux-ci. AWS IoT

Matériel requis :

- Votre environnement de développement et de test local décrit dans la section précédente
- Le Raspberry Pi que vous avez utilisé dans la section précédente
- La carte mémoire microSD du Raspberry Pi que vous avez utilisée dans la section précédente

Procédures dans ce didacticiel

- Préparez le Raspberry Pi pour démontrer la communication de messages MQTT
- Démonstration de la publication de messages avec le AWS IoT Device Client
- Démontrer l'abonnement aux messages avec le AWS IoT Device Client

Préparez le Raspberry Pi pour démontrer la communication de messages MQTT

Cette procédure crée les ressources dans AWS IoT et dans le Raspberry Pi pour démontrer la communication de messages MQTT à l'aide du AWS IoT Device Client.

Procédures décrites dans cette section :

- Créez les fichiers de certificat pour démontrer la communication MQTT
- Provisionnez votre appareil pour démontrer la communication MQTT
- Configurer le fichier de configuration du client du AWS IoT périphérique et le client de test MQTT pour démontrer la communication MQTT

Créez les fichiers de certificat pour démontrer la communication MQTT

Cette procédure crée les fichiers de certificat de l'appareil pour cette démonstration.

Pour créer et télécharger les fichiers de certificat d'appareil pour votre Raspberry Pi

1. Dans la fenêtre du terminal de votre ordinateur hôte local, entrez ces commandes pour créer les fichiers de certificat de périphérique pour votre appareil.

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

   La commande renvoie une réponse comme celle-ci. Enregistrez la valeur *certificateArn* pour une utilisation ultérieure.

   ```
   {
   "certificateArn": "arn:aws:iot:us-
   west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
   "certificateId":
    "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
   "certificatePem": "-----BEGIN CERTIFICATE-----
   \nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
   \n",
   "keyPair": {
   ```

```
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. Entrez les commandes suivantes pour définir les autorisations sur le répertoire des certificats et ses fichiers.

```
chmod 700 ~/certs/pubsub
chmod 644 ~/certs/pubsub/*
chmod 600 ~/certs/pubsub/private.pem.key
```

3. Exécutez cette commande pour vérifier les autorisations sur vos répertoires et fichiers de certificats.

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

   Le résultat de la commande doit être identique à ce que vous voyez ici, sauf que les dates et heures du fichier seront différentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

4. Entrez ces commandes pour créer les répertoires des fichiers journaux.

```
mkdir ~/.aws-iot-device-client
mkdir ~/.aws-iot-device-client/log
chmod 745 ~/.aws-iot-device-client/log
echo " " > ~/.aws-iot-device-client/log/aws-iot-device-client.log
echo " " > ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
chmod 600 ~/.aws-iot-device-client/log/*
```

Provisionnez votre appareil pour démontrer la communication MQTT

Cette section crée les AWS IoT ressources qui approvisionnent votre Raspberry Pi AWS IoT.

Pour approvisionner votre appareil en AWS IoT :

1. Dans la fenêtre du terminal de votre ordinateur hôte local, entrez la commande suivante pour obtenir l'adresse du point de terminaison des données de l'appareil pour votre Compte AWS.

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

   La valeur du point de terminaison n'a pas changé depuis l'exécution de cette commande pour le didacticiel précédent. La réexécution de la commande ici est effectuée pour faciliter la recherche et le collage de la valeur du point de terminaison des données dans le fichier de configuration utilisé dans ce didacticiel.

   La commande des étapes précédentes renvoie une réponse comme celle-ci. Enregistrez la valeur *endpointAddress* pour une utilisation ultérieure.

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Entrez cette commande pour créer une nouvelle ressource d' AWS IoT objets pour votre Raspberry Pi.

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

   Comme une AWS IoT ressource d'objets est une représentation virtuelle de votre appareil dans le cloud, nous pouvons créer plusieurs ressources d'objets AWS IoT à utiliser à différentes fins. Ils peuvent tous être utilisés par le même appareil IoT physique pour représenter différents aspects de l'appareil.

   Ces didacticiels n'utiliseront qu'une seule ressource à la fois pour représenter le Raspberry Pi. Ainsi, dans ces didacticiels, ils représentent les différentes démos. Ainsi, après avoir créé les AWS IoT ressources pour une démonstration, vous pouvez revenir en arrière et répéter la démonstration en utilisant les ressources que vous avez créées spécifiquement pour chacune d'elles.

   Si votre ressource d' AWS IoT objet a été créée, la commande renvoie une réponse comme celle-ci.

   ```
   {
   ```

```
"thingName": "PubSubTestThing",
"thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/PubSubTestThing",
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Dans la fenêtre du terminal :

   a. Ouvrez un éditeur de texte comme nano.

   b. Copiez ce document JSON et collez-le dans votre éditeur de texte ouvert.

      JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
      ]
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
      ]
    }
  ]
}
```

c. Dans l'éditeur, dans chaque `Resource` section du document de politique, remplacez *us-west-2:57EXAMPLE833* par votre Région AWS caractère deux-points (:)) et votre Compte AWS numéro à 12 chiffres.

d. Enregistrez le fichier dans votre éditeur de texte sous le nom **~/policies/pubsub_test_thing_policy.json**.

4. Exécutez cette commande pour utiliser le document de stratégie décrit dans les étapes précédentes afin de créer une AWS IoT stratégie.

```
aws iot create-policy \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_test_thing_policy.json"
```

Si la politique est créée, la commande renvoie une réponse comme celle-ci.

JSON

```
{
    "policyName": "PubSubTestThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
PubSubTestThingPolicy",
    "policyDocument": "{\n\"Version\": \"2012-10-17\",\n\"Statement\": [\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},
\n{\n\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource
\": [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},
\n{\n\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource
\": [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic
\"\n]\n},\n{\n\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Receive\"\n],
```

```
\n\"Resource\": [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*
\"\n]\n}\n]\n}\n",
    "policyVersionId": "1"
```

5. Exécutez cette commande pour associer la politique au certificat de l'appareil. Remplacez *certificateArn* par la valeur `certificateArn` que vous avez enregistrée précédemment dans cette section.

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

   Si elle aboutit, cette commande ne renvoie rien.

6. Exécutez cette commande pour associer le certificat de l'appareil à la ressource AWS IoT d'objet. Remplacez *certificateArn* par la valeur `certificateArn` que vous avez enregistrée précédemment dans cette section.

   ```
   aws iot attach-thing-principal \
   --thing-name "PubSubTestThing" \
   --principal "certificateArn"
   ```

   Si elle aboutit, cette commande ne renvoie rien.

Une fois que vous avez correctement configuré votre appareil AWS IoT, vous êtes prêt à continuerthe section called “Configurer le fichier de configuration du client du périphérique et le client MQTT”.

Configurer le fichier de configuration du client du AWS IoT périphérique et le client de test MQTT pour démontrer la communication MQTT

Cette procédure crée un fichier de configuration pour tester le client du AWS IoT périphérique.

Pour créer le fichier de configuration permettant de tester le client du AWS IoT périphérique

1. Dans la fenêtre du terminal sur votre ordinateur hôte local connecté à votre Raspberry Pi :

   a. Ouvrez un éditeur de texte comme `nano`.

   b. Copiez ce document JSON et collez-le dans votre éditeur de texte ouvert.

   ```
   {
   ```

```
"endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
"cert": "~/certs/pubsub/device.pem.crt",
"key": "~/certs/pubsub/private.pem.key",
"root-ca": "~/certs/AmazonRootCA1.pem",
"thing-name": "PubSubTestThing",
"logging": {
  "enable-sdk-logging": true,
  "level": "DEBUG",
  "type": "STDOUT",
  "file": ""
},
"jobs": {
  "enabled": false,
  "handler-directory": ""
},
"tunneling": {
  "enabled": false
},
"device-defender": {
  "enabled": false,
  "interval": 300
},
"fleet-provisioning": {
  "enabled": false,
  "template-name": "",
  "template-parameters": "",
  "csr-file": "",
  "device-key": ""
},
"samples": {
  "pub-sub": {
    "enabled": true,
    "publish-topic": "test/dc/pubtopic",
    "publish-file": "",
    "subscribe-topic": "test/dc/subtopic",
    "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
  }
},
"config-shadow": {
  "enabled": false
},
"sample-shadow": {
  "enabled": false,
  "shadow-name": "",
```

```
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. Remplacez la *endpoint* valeur par le point de terminaison de données de l'appareil dans Compte AWS lequel vous l'avez trouvéthe section called “Approvisionnez votre appareil en AWS IoT Core”.

d. Enregistrez le fichier dans votre éditeur de texte sous le nom **~/dc-configs/dc-pubsub-config.json**.

e. Exécutez cette commande pour définir les autorisations sur le nouveau fichier de configuration.

```
chmod 644 ~/dc-configs/dc-pubsub-config.json
```

2. Pour préparer le client de test MQTT à s'abonner à tous les messages MQTT :

a. Sur votre ordinateur hôte local, dans la AWS IoT console, choisissez le client de test MQTT.

b. Dans l'onglet S'abonner à une rubrique, dans le filtre de rubrique, entrez **#** (un seul signe dièse), puis choisissez S'abonner.

c. Sous l'étiquette Abonnements, confirmez que vous voyez **#** (un seul signe dièse).

Laissez la fenêtre avec le client de test MQTT ouverte pendant que vous poursuivez ce didacticiel.

Après avoir enregistré le fichier et configuré le client de test MQTT, vous êtes prêt à continuer the section called “Publiez des messages avec IoT Device Client”.

Démonstration de la publication de messages avec le AWS IoT Device Client

Les procédures décrites dans cette section montrent comment le client du AWS IoT périphérique peut envoyer des messages MQTT par défaut et personnalisés.

Les déclarations de politique que vous avez créées à l'étape précédente pour ces exercices autorisent le Raspberry Pi à effectuer les actions suivantes :

- **iot:Connect**

Permet au client nomméPubSubTestThing, votre Raspberry Pi exécutant le AWS IoT Device Client, de se connecter.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Connect"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
  ]
}
```

- **iot:Publish**

  Permet au Raspberry Pi de publier des messages dont la rubrique MQTT est `test/dc/pubtopic`.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Publish"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
  ]
}
```

  L'action `iot:Publish` donne l'autorisation de publier dans les rubriques MQTT répertoriées dans le tableau de ressources. Le contenu de ces messages n'est pas contrôlé par la déclaration de politique.

Publiez le message par défaut à l'aide du AWS IoT Device Client

Cette procédure exécute le client de AWS IoT périphérique afin qu'il publie un seul message MQTT par défaut que le client de test MQTT reçoit et affiche.

Pour envoyer le message MQTT par défaut depuis le client du AWS IoT périphérique

1. Assurez-vous que la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi et la fenêtre avec le client de test MQTT sont visibles pendant que vous effectuez cette procédure.
2. Dans la fenêtre du terminal, entrez ces commandes pour exécuter le client de AWS IoT périphérique à l'aide du fichier de configuration créé dansthe section called “Créez le fichier de configuration”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-config.json
   ```

   Dans la fenêtre du terminal, le client du AWS IoT périphérique affiche les messages d'information et les éventuelles erreurs survenant lors de son exécution.

   Si aucune erreur n'est affichée dans la fenêtre du terminal, passez en revue le client de test MQTT.
3. Dans le client de test MQTT, dans la fenêtre Subscriptions(Abonnements), consultez le Hello World ! message envoyé à la rubrique `test/dc/pubtopic`.
4. Si le client de l' AWS IoT appareil n'affiche aucune erreur et que vous voyez Hello World ! envoyé au `test/dc/pubtopic` message dans le client de test MQTT, vous avez démontré une connexion réussie.
5. Dans la fenêtre du terminal, entrez **^C** (Ctrl-C) pour arrêter le AWS IoT Device Client.

Après avoir démontré que le AWS IoT Device Client a publié le message MQTT par défaut, vous pouvez passer authe section called “Publier un message MQTT personnalisé”.

Publier un message personnalisé à l'aide du AWS IoT Device Client

Les procédures décrites dans cette section créent un message MQTT personnalisé, puis exécutent le client de périphérique AWS IoT afin qu'il publie le message MQTT personnalisé une fois pour que le client de test MQTT le reçoive et l'affiche.

Création d'un message MQTT personnalisé pour le client du AWS IoT périphérique

Effectuez ces étapes dans la fenêtre du terminal sur l'ordinateur hôte local connecté à votre Raspberry Pi.

Pour créer un message personnalisé à publier par le AWS IoT Device Client

1. Dans la fenêtre du terminal, ouvrez un éditeur de texte comme `nano`.
2. Dans l'éditeur de texte, copiez et collez le document JSON suivant. Il s'agira de la charge utile du message MQTT que le client du AWS IoT périphérique publiera.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

3. Enregistrement du contenu de l'éditeur de texte avec **`~/messages/sample-ws-message.json`**.
4. Saisissez la commande suivante pour définir les autorisations du fichier de messages que vous venez de créer.

```
chmod 600 ~/messages/*
```

Pour créer un fichier de configuration que le client du AWS IoT périphérique utilisera pour envoyer le message personnalisé

1. Dans la fenêtre du terminal, dans un éditeur de texte tel que`nano`, ouvrez le fichier de configuration du client de AWS IoT périphérique existant :**`~/dc-configs/dc-pubsub-config.json`**.
2. Modifiez l'objet `samples` pour qu'il ressemble à ceci. Aucune autre partie de ce fichier n'a besoin d'être modifiée.

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/subtopic",
```

```
        "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

3. Enregistrement du contenu de l'éditeur de texte avec **`~/dc-configs/dc-pubsub-custom-config.json`**.
4. Exécutez cette commande pour définir les autorisations sur le nouveau fichier de configuration.

```
chmod 644 ~/dc-configs/dc-pubsub-custom-config.json
```

Publiez le message MQTT personnalisé à l'aide du AWS IoT Device Client

Cette modification n'affecte que le contenu de la charge utile des messages MQTT, de sorte que la politique actuelle continuera de fonctionner. Toutefois, si le sujet MQTT (tel que défini par la valeur `publish-topic` dans`~/dc-configs/dc-pubsub-custom-config.json`) était modifié, la déclaration de politique `iot::Publish` devrait également être modifiée pour permettre au Raspberry Pi de publier sur le nouveau sujet MQTT.

Pour envoyer le message MQTT depuis le client du AWS IoT périphérique

1. Assurez-vous que la fenêtre du terminal et la fenêtre avec le client de test MQTT sont visibles pendant que vous effectuez cette procédure. Assurez-vous également que votre client de test MQTT est toujours abonné au filtre # topic. Si ce n'est pas le cas, abonnez-vous à nouveau au filtre de rubrique #.
2. Dans la fenêtre du terminal, entrez ces commandes pour exécuter AWS IoT Device Client à l'aide du fichier de configuration créé dans the section called “Créez le fichier de configuration”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   Dans la fenêtre du terminal, le client du AWS IoT périphérique affiche les messages d'information et les éventuelles erreurs survenant lors de son exécution.

   Si aucune erreur n'est affichée dans la fenêtre du terminal, passez en revue le client de test MQTT.
3. Dans le client de test MQTT, dans la fenêtre Abonnements, consultez la charge utile du message personnalisé envoyé à la rubrique du message `test/dc/pubtopic`.

4. Si le AWS IoT Device Client n'affiche aucune erreur et que vous voyez la charge utile du message personnalisé que vous avez publié sur le `test/dc/pubtopic` message dans le client de test MQTT, cela signifie que vous avez publié un message personnalisé avec succès.
5. Dans la fenêtre du terminal, entrez **^C** (Ctrl-C) pour arrêter le AWS IoT Device Client.

Après avoir démontré que le AWS IoT Device Client a publié une charge utile de messages personnalisée, vous pouvez continuerthe section called “Abonnez-vous aux messages avec IoT Device Client”.

Démontrer l'abonnement aux messages avec le AWS IoT Device Client

Dans cette section, vous allez présenter deux types d'abonnements aux messages :

- Abonnement thématique unique
- Abonnement à un sujet générique

Ces déclarations de politique dans la politique créée pour ces exercices autorisent le Raspberry Pi à effectuer ces actions :

- **`iot:Receive`**

  Permet au AWS IoT Device Client de recevoir des sujets MQTT qui correspondent à ceux nommés dans l'`Resource`objet.

  ```
  {
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
    ]
  }
  ```

- **`iot:Subscribe`**

  Permet au AWS IoT Device Client de s'abonner aux filtres de sujets MQTT qui correspondent à ceux nommés dans l'`Resource`objet.

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
      ]
    }
```

Abonnez-vous à un seul sujet de message MQTT

Cette procédure montre comment le client du AWS IoT périphérique peut s'abonner aux messages MQTT et les enregistrer.

Dans la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi, listez le contenu du fichier **~/dc-configs/dc-pubsub-custom-config.json** ou ouvrez-le dans un éditeur de texte pour en vérifier le contenu. Localisez l'objet `samples`, qui devrait ressembler à ceci.

```
"samples": {
  "pub-sub": {
    "enabled": true,
    "publish-topic": "test/dc/pubtopic",
    "publish-file": "~/messages/sample-ws-message.json",
    "subscribe-topic": "test/dc/subtopic",
    "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

Notez que la valeur `subscribe-topic` est la rubrique MQTT auquel le AWS IoT Device Client s'abonnera lors de son exécution. Le client du AWS IoT périphérique écrit les charges utiles des messages qu'il reçoit dans le cadre de cet abonnement dans le fichier nommé dans la `subscribe-file` valeur.

Pour s'abonner à un sujet de message MQTT depuis le AWS IoT Device Client

1. Assurez-vous que la fenêtre du terminal et la fenêtre avec le client de test MQTT sont visibles pendant que vous effectuez cette procédure. Assurez-vous également que votre client de test MQTT est toujours abonné au filtre # topic. Si ce n'est pas le cas, abonnez-vous à nouveau au filtre de rubrique #.

2. Dans la fenêtre du terminal, entrez ces commandes pour exécuter le client de AWS IoT périphérique à l'aide du fichier de configuration créé dansthe section called “Créez le fichier de configuration”.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
   ```

   Dans la fenêtre du terminal, le client du AWS IoT périphérique affiche les messages d'information et les éventuelles erreurs survenant lors de son exécution.

   Si aucune erreur ne s'affiche dans la fenêtre du terminal, continuez dans la AWS IoT console.

3. Dans la AWS IoT console, dans le client de test MQTT, choisissez l'onglet Publier dans une rubrique.
4. Dans Topic name (Nom de la rubrique), saisissez **`test/dc/subtopic`**.
5. Dans Charge utile du message , passez en revue le contenu du message.
6. Choisissez Publish pour publier le message MQTT.
7. Dans la fenêtre du terminal, recherchez le message reçu du client du AWS IoT périphérique qui ressemble à ceci.

   ```
   2021-11-10T16:02:20.890Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 45 bytes
   ```

8. Après avoir vu le message reçu indiquant que le message a été reçu, entrez **`^C`** (Ctrl-C) pour arrêter le client du AWS IoT périphérique.
9. Entrez cette commande pour afficher la fin du fichier journal des messages et voir le message que vous avez publié à partir du client de test MQTT.

   ```
   tail ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
   ```

En visualisant le message dans le fichier journal, vous avez démontré que le client du périphérique AWS IoT a reçu le message que vous avez publié à partir du client de test MQTT.

Abonnez-vous à plusieurs sujets de message MQTT en utilisant des caractères génériques

Ces procédures montrent comment le client du AWS IoT périphérique peut s'abonner à des messages MQTT et les enregistrer à l'aide de caractères génériques. Pour ce faire, vous allez :

1. Mettez à jour le filtre de rubrique utilisé par le AWS IoT Device Client pour s'abonner aux rubriques MQTT.
2. Mettez à jour la politique utilisée par l'appareil pour autoriser les nouveaux abonnements.
3. Exécutez le AWS IoT Device Client et publiez des messages depuis la console de test MQTT.

Pour créer un fichier de configuration permettant de s'abonner à plusieurs rubriques de messages MQTT à l'aide d'un filtre MQTT générique

1. Dans la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi, ouvrez **`~/dc-configs/dc-pubsub-custom-config.json`** pour le modifier et localisez l'objet `samples`.
2. Dans l'éditeur de texte, localisez l'`samples`objet et mettez à jour la valeur `subscribe-topic` pour qu'elle ressemble à ceci.

   ```
   "samples": {
     "pub-sub": {
       "enabled": true,
       "publish-topic": "test/dc/pubtopic",
       "publish-file": "~/messages/sample-ws-message.json",
       "subscribe-topic": "test/dc/#",
       "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
   ```

   La nouvelle valeur `subscribe-topic` est un filtre thématique MQTT avec un caractère générique MQTT à la fin. Ceci décrit un abonnement à tous les sujets MQTT commençant par `test/dc/`. Le client du AWS IoT périphérique écrit les charges utiles des messages qu'il reçoit dans le cadre de cet abonnement dans le fichier indiqué dans`subscribe-file`.
3. Enregistrez le fichier de configuration modifié sous **`~/dc-configs/dc-pubsub-wild-config.json`**, et quittez l'éditeur.

Pour modifier la politique utilisée par votre Raspberry Pi afin d'autoriser l'abonnement et la réception de plusieurs rubriques de messages MQTT

1. Dans la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi, dans votre éditeur de texte préféré, ouvrez **`~/policies/pubsub_test_thing_policy.json`** pour modification, puis recherchez les `iot::Subscribe` et les déclarations de politique `iot::Receive` dans le fichier.

2. Dans la déclaration de politique `iot::Subscribe`, mettez à jour la chaîne de l'objet Resource `subtopic` à remplacer par `*`, afin qu'elle ressemble à ceci.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*"
  ]
}
```

> **Note**
>
> Les caractères génériques du filtre thématique MQTT sont le + (signe plus) et le # (signe dièse). Une demande d'abonnement avec un # à la fin permet de souscrire à tous les rubriques commençant par la chaîne qui précède le caractère # (par exemple, `test/dc/` dans ce cas).
> La valeur de la ressource indiquée dans la déclaration de politique qui autorise cet abonnement doit toutefois utiliser un * (astérisque) à la place du # (signe dièse) dans l'ARN du filtre de rubrique. Cela est dû au fait que le processeur de politiques utilise un caractère générique différent de celui utilisé par MQTT.
> Pour plus d'informations sur l'utilisation de caractères génériques pour les rubriques et de filtres de rubriques dans les politiques, consultez Utilisation de caractères génériques dans MQTT et les politiques AWS IoT Core.

3. Dans la déclaration de politique `iot::Receive`, mettez à jour la chaîne de l'objet Resource `subtopic` à remplacer par `*`, afin qu'elle ressemble à ceci.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*"
  ]
}
```

4. Enregistrez le document de politique mis à jour et quittez l'éditeur **~/policies/pubsub_wild_test_thing_policy.json**.
5. Entrez cette commande pour mettre à jour la politique de ce didacticiel afin d'utiliser les nouvelles définitions de ressources.

```
aws iot create-policy-version \
--set-as-default \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_wild_test_thing_policy.json"
```

Si la commande réussit, elle renvoie une réponse comme celle-ci. Notez que `policyVersionId` est maintenant 2, ce qui indique qu'il s'agit de la deuxième version de cette politique.

Si vous avez correctement mis à jour la politique, vous pouvez passer à la procédure suivante.


```
{
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
    "policyDocument": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement\": [\n
 {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Connect\"\n
    ],\n      \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:client/
PubSubTestThing\"\n      ]\n    },\n    {\n      \"Effect\": \"Allow\",\n
   \"Action\": [\n        \"iot:Publish\"\n      ],\n      \"Resource\": [\n
        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n
  ]\n    },\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n
 \"iot:Subscribe\"\n      ],\n      \"Resource\": [\n        \"arn:aws:iot:us-
west-2:57EXAMPLE833:topicfilter/test/dc/*\"\n      ]\n    },\n    {\n      \"Effect
\": \"Allow\",\n      \"Action\": [\n        \"iot:Receive\"\n      ],\n
 \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*\"\n
     ]\n    }\n  ]\n}\n",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```


Si vous recevez un message d'erreur indiquant qu'il existe trop de versions de politique pour en enregistrer une nouvelle, entrez cette commande pour répertorier les versions actuelles de la politique. Consultez la liste renvoyée par cette commande pour rechercher une version de politique que vous pouvez supprimer.

```
aws iot list-policy-versions --policy-name "PubSubTestThingPolicy"
```

Saisissez cette commande pour supprimer une version dont vous n'avez plus besoin. Notez que vous ne pouvez pas supprimer la version de politique par défaut. La version de politique par défaut est celle dont la valeur `isDefaultVersion` est de `true`.

```
aws iot delete-policy-version \
--policy-name "PubSubTestThingPolicy" \
--policy-version-id policyId
```

Après avoir supprimé une version de politique, réessayez cette étape.

Avec le fichier de configuration et la politique mis à jour, vous êtes prêt à démontrer les abonnements génériques avec le AWS IoT Device Client.

Pour montrer comment le client du AWS IoT périphérique s'abonne à plusieurs sujets de messages MQTT et en reçoit

1. Dans le client de test MQTT, vérifiez les abonnements. Si le client de test MQTT est abonné au filtre de rubrique **#**, passez à l'étape suivante. Sinon, dans le client de test MQTT, dans l'onglet S'abonner à une rubrique, dans le filtre de rubrique, entrez **#** (un signe dièse), puis choisissez S'abonner pour vous y abonner.
2. Dans la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi, entrez ces commandes pour démarrer le AWS IoT Device Client.

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-wild-config.json
   ```

3. Tout en regardant la sortie du AWS IoT Device Client dans la fenêtre du terminal de l'ordinateur hôte local, retournez au client de test MQTT. Dans l'onglet Publier dans une rubrique, dans Nom de rubrique, entrez **`test/dc/subtopic`**, puis choisissez Publier.
4. Dans la fenêtre du terminal, vérifiez que le message a bien été reçu en recherchant un message tel que :

   ```
   2021-11-10T16:34:20.101Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 76 bytes
   ```

5. Tout en regardant la sortie du AWS IoT Device Client dans la fenêtre du terminal de l'ordinateur hôte local, retournez au client de test MQTT. Dans l'onglet Publier dans une rubrique, dans Nom de rubrique, entrez **`test/dc/subtopic2`**, puis choisissez Publier.
6. Dans la fenêtre du terminal, vérifiez que le message a bien été reçu en recherchant un message tel que :

   ```
   2021-11-10T16:34:32.078Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 77 bytes
   ```
7. Après avoir vu les messages confirmant la réception des deux messages, entrez **`^C`** (Ctrl-C) pour arrêter le AWS IoT Device Client.
8. Entrez cette commande pour afficher la fin du fichier journal des messages et voir le message que vous avez publié à partir du client de test MQTT.

   ```
   tail -n 20 ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
   ```

   > **Note**
   >
   > Le fichier journal contient uniquement les charges utiles des messages. Les sujets des messages ne sont pas enregistrés dans le fichier journal des messages reçus.
   > Vous pouvez également voir le message publié par le AWS IoT Device Client dans le journal reçu. En effet, le filtre de sujet générique inclut ce sujet de message et, parfois, la demande d'abonnement peut être traitée par l’agent de messages avant que le message publié ne soit envoyé aux abonnés.

Les entrées du fichier journal indiquent que les messages ont été reçus. Vous pouvez répéter cette procédure en utilisant d'autres noms de rubrique. Tous les messages dont le nom de rubrique commence par `test/dc/` doivent être reçus et enregistrés. Les messages dont le nom de rubrique commence par un autre texte sont ignorés.

Après avoir démontré comment le AWS IoT Device Client peut publier des messages MQTT et s'y abonner, passez àDidacticiel : Démonstration d'actions à distance (jobs) avec AWS IoT Device Client.

Didacticiel : Démonstration d'actions à distance (jobs) avec AWS IoT Device Client

Dans ces didacticiels, vous allez configurer et déployer des tâches sur votre Raspberry Pi pour montrer comment envoyer des opérations à distance à vos appareils IoT.

Pour démarrer ce didacticiel :

- Configurez un Raspberry Pi sur votre ordinateur hôte local comme indiqué dans la section précédente.
- Si vous n'avez pas terminé le didacticiel de la section précédente, vous pouvez essayer ce didacticiel en utilisant le Raspberry Pi avec une carte microSD contenant l'image que vous avez enregistrée après avoir installé le client du AWS IoT périphérique. (Facultatif) Enregistrez l'image de la carte microSD
- Si vous avez déjà lancé cette démo, vérifiez si vous ??? souhaitez supprimer toutes les AWS IoT ressources que vous avez créées lors des exécutions précédentes afin d'éviter les erreurs liées aux ressources dupliquées.

Ce didacticiel vous prendra environ 45 minutes.

Lorsque vous avez terminé avec cette rubrique :

- Vous aurez démontré les différentes manières dont votre appareil IoT peut utiliser le AWS IoT Core pour exécuter des opérations à distance gérées par AWS IoT .

Matériel requis :

- Votre environnement de développement et de test local dans lequel vous avez testé dans la section précédente
- Le Raspberry Pi que vous avez testé dans la section précédente
- La carte mémoire microSD du Raspberry Pi que vous avez testée dans la section précédente

Procédures dans ce didacticiel

- Préparez le Raspberry Pi pour exécuter des tâches
- Créez et exécutez le job AWS IoT avec AWS IoT Device Client

Préparez le Raspberry Pi pour exécuter des tâches

Les procédures décrites dans cette section décrivent comment préparer votre Raspberry Pi à exécuter des tâches à l'aide du AWS IoT Device Client.

> **Note**
>
> Ces procédures sont spécifiques à l'appareil. Si vous souhaitez exécuter les procédures décrites dans cette section avec plusieurs appareils à la fois, chaque appareil aura besoin de sa propre politique, d'un certificat et d'un nom d'objet uniques et spécifiques à l'appareil. Pour attribuer à chaque appareil ses ressources uniques, effectuez cette procédure une fois pour chaque appareil tout en modifiant les éléments spécifiques au appareil comme décrit dans les procédures.

Procédures dans ce didacticiel

- Provisionnez votre Raspberry Pi pour présenter des jobs
- Configurer le AWS IoT Device Client pour exécuter l'agent de tâches

Provisionnez votre Raspberry Pi pour présenter des jobs

Les procédures décrites dans cette section approvisionnent votre Raspberry Pi en AWS IoT créant AWS IoT des ressources et des certificats d'appareil pour celui-ci.

Rubriques

- Créez et téléchargez des fichiers de certificat d'appareil pour illustrer les AWS IoT tâches
- Créer des AWS IoT ressources pour présenter des AWS IoT offres d'emploi

Créez et téléchargez des fichiers de certificat d'appareil pour illustrer les AWS IoT tâches

Cette procédure crée les fichiers de certificat de l'appareil pour cette démonstration.

Si vous préparez plusieurs appareils, cette procédure doit être exécutée sur chaque appareil.

Pour créer et télécharger les fichiers de certificat d'appareil pour votre Raspberry Pi :

Dans la fenêtre du terminal sur votre ordinateur hôte local connecté à votre Raspberry Pi :

1. Entrez la commande suivante pour créer les fichiers de certificat de l'appareil pour votre appareil.

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

La commande renvoie une réponse comme celle-ci. Enregistrez la valeur *certificateArn* pour une utilisation ultérieure.

```
{
"certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----
\n",
"keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----
\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. Entrez les commandes suivantes pour définir les autorisations sur le répertoire des certificats et ses fichiers.

```
chmod 700 ~/certs/jobs
chmod 644 ~/certs/jobs/*
chmod 600 ~/certs/jobs/private.pem.key
```

3. Exécutez cette commande pour vérifier les autorisations sur vos répertoires et fichiers de certificats.

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

Le résultat de la commande doit être identique à ce que vous voyez ici, sauf que les dates et heures du fichier seront différentes.

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

Après avoir téléchargé les fichiers de certificat de l'appareil sur votre Raspberry Pi, vous êtes prêt à continuer the section called “Fournir un Raspberry Pi pour les tâches”.

Créer des AWS IoT ressources pour présenter des AWS IoT offres d'emploi

Créez les AWS IoT ressources pour cet appareil.

Si vous préparez plusieurs appareils, cette procédure doit être exécutée pour chaque appareil.

Pour approvisionner votre appareil en AWS IoT :

Dans la fenêtre du terminal sur votre ordinateur hôte local connecté à votre Raspberry Pi :

1. Saisissez la commande suivante pour obtenir l'adresse du point de terminaison de données de l'appareil pour votre Compte AWS.

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

   La valeur du point de terminaison n'a pas changé depuis la dernière exécution de cette commande. La réexécution de la commande ici facilite la recherche et le collage de la valeur du point de terminaison de données dans le fichier de configuration utilisé dans ce didacticiel.

   La commande describe-endpoint renvoie une réponse comme celle-ci. Enregistrez la valeur *endpointAddress* pour une utilisation ultérieure.

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. *uniqueThingName*Remplacez-le par un nom unique pour votre appareil. Si vous souhaitez exécuter ce didacticiel avec plusieurs appareils, attribuez à chaque appareil son propre nom. Par exemple, **TestDevice01**, **TestDevice02**, et ainsi de suite.

   Entrez cette commande pour créer une nouvelle ressource d' AWS IoT objets pour votre Raspberry Pi.

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

Comme une AWS IoT ressource d'objets est une représentation virtuelle de votre appareil dans le cloud, nous pouvons créer plusieurs ressources d'objets AWS IoT à utiliser à différentes fins. Ils peuvent tous être utilisés par le même appareil IoT physique pour représenter différents aspects de l'appareil.

> **Note**
>
> Lorsque vous souhaitez sécuriser la politique pour plusieurs appareils, vous pouvez utiliser `${iot:Thing.ThingName}` à la place du nom d'objet statique *uniqueThingName*.

Ces didacticiels n'utiliseront qu'une seule ressource à la fois par appareil. Ainsi, dans ces didacticiels, ils représentent les différentes démos. Ainsi, après avoir créé les AWS IoT ressources pour une démonstration, vous pouvez revenir en arrière et répéter les démos en utilisant les ressources que vous avez créées spécifiquement pour chacune d'elles.

Si votre ressource d' AWS IoT objet a été créée, la commande renvoie une réponse comme celle-ci. Enregistrez la valeur *thingArn* pour une utilisation ultérieure lorsque vous créerez le job à exécuter sur cet appareil.

```
{
"thingName": "uniqueThingName",
"thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/uniqueThingName",
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. Dans la fenêtre du terminal :

   a. Ouvrez un éditeur de texte comme `nano`.

   b. Copiez ce document JSON et collez-le dans votre éditeur de texte ouvert.

      JSON

      ```
      {
      ```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/uniqueThingName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/job/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/
jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/
things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/
subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/
jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/
things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
```

```
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/
things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeJobExecution",
        "iot:GetPendingJobExecutions",
        "iot:StartNextPendingJobExecution",
        "iot:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/
things/uniqueThingName"
      ]
    }
  ]
}
```

c. Dans l'éditeur, dans la `Resource` section de chaque déclaration de politique, remplacez *us-west-2:57EXAMPLE833* par le vôtre Région AWS, le caractère deux-points (:)) et votre Compte AWS numéro à 12 chiffres.

d. Dans l'éditeur, dans chaque déclaration de politique, remplacez-le *uniqueThingName* par le nom que vous avez donné à cette ressource.

e. Enregistrez le fichier dans votre éditeur de texte sous le nom **~/policies/jobs_test_thing_policy.json**

   Si vous exécutez cette procédure pour plusieurs appareils, enregistrez le fichier sous ce nom sur chaque appareil.

4. *uniqueThingName*Remplacez-le par le nom de l'objet du périphérique, puis exécutez cette commande pour créer une AWS IoT politique adaptée à cet appareil.

```
aws iot create-policy \
--policy-name "JobTestPolicyForuniqueThingName" \
--policy-document "file://~/policies/jobs_test_thing_policy.json"
```

Si la politique est créée, la commande renvoie une réponse comme celle-ci.

JSON

```
{
    "policyName": "JobTestPolicyFor uniqueThingName",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
JobTestPolicyFor uniqueThingName",
    "policyDocument": "{\n\"Version\": \"2012-10-17\",\n\"Statement\": [\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\":
 [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},
\n{\n\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource
\": [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},
\n{\n\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource
\": [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic
\"\n]\n},\n{\n\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Receive\"\n],
\n\"Resource\": [\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*
\"\n]\n}\n]\n}\n",
    "policyVersionId": "1"
```

5. Remplacez-le *uniqueThingName* par le nom de l'objet du périphérique et *certificateArn* par la `certificateArn` valeur que vous avez enregistrée précédemment dans cette section pour ce périphérique, puis exécutez cette commande pour associer la politique au certificat de l'appareil.

   ```
   aws iot attach-policy \
   --policy-name "JobTestPolicyFor uniqueThingName" \
   --target "certificateArn"
   ```

   Si elle aboutit, cette commande ne renvoie rien.

6. Remplacez *uniqueThingName* par le nom de l'objet pour le périphérique, remplacez *certificateArn* par la `certificateArn` valeur que vous avez enregistrée précédemment dans cette section, puis exécutez cette commande pour associer le certificat du périphérique à la ressource de l' AWS IoT objet.

   ```
   aws iot attach-thing-principal \
   --thing-name "uniqueThingName" \
   --principal "certificateArn"
   ```

   Si elle aboutit, cette commande ne renvoie rien.

Une fois que vous avez correctement configuré votre Raspberry Pi, vous êtes prêt à répéter cette section pour un autre Raspberry Pi lors de votre test ou, si tous les appareils ont été configurés, continuer à the section called “Configurer Device Client pour exécuter des tâches”.

Configurer le AWS IoT Device Client pour exécuter l'agent de tâches

Cette procédure crée un fichier de configuration pour que le client du AWS IoT périphérique exécute l'agent de tâches :.

Remarque : si vous préparez plusieurs appareils, cette procédure doit être effectuée sur chaque appareil.

Pour créer le fichier de configuration permettant de tester le client du AWS IoT périphérique :

1. Dans la fenêtre du terminal sur votre ordinateur hôte local connecté à votre Raspberry Pi :

   a. Ouvrez un éditeur de texte comme nano.

   b. Copiez ce document JSON et collez-le dans votre éditeur de texte ouvert.

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/jobs/device.pem.crt",
  "key": "~/certs/jobs/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "uniqueThingName",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": true,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
```

```
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": false,
      "publish-topic": "",
      "publish-file": "",
      "subscribe-topic": "",
      "subscribe-file": ""
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. Remplacez la *endpoint* valeur par la valeur du point de terminaison des données de l'appareil Compte AWS que vous avez trouvée dansthe section called “Approvisionnez votre appareil en AWS IoT Core”.

d. *uniqueThingName*Remplacez-le par le nom de l'objet que vous avez utilisé pour cet appareil.

e. Enregistrez le fichier dans votre éditeur de texte sous le nom **~/dc-configs/dc-jobs-config.json**

2. Exécutez cette commande pour définir les autorisations de fichier du nouveau fichier de configuration.

```
chmod 644 ~/dc-configs/dc-jobs-config.json
```

Vous n'utiliserez pas le client de test MQTT pour cette rubrique. Alors que l'appareil échange des messages MQTT relatifs aux tâches AWS IoT, les messages de progression des tâches ne sont

échangés qu'avec l'appareil exécutant la tâche. Les messages de progression des tâches étant échangés uniquement avec l'appareil exécutant la tâche, vous ne pouvez pas vous y abonner depuis un autre appareil, tel que la AWS IoT console.

Après avoir enregistré le fichier de configuration, vous êtes prêt à continuer the section called "Création et exécution d'une tâche dans le domaine de l'IoT".

Créez et exécutez le job AWS IoT avec AWS IoT Device Client

Les procédures décrites dans cette section créent un document de travail et une ressource de AWS IoT travail. Une fois que vous avez créé la ressource de tâche, AWS IoT envoie le document de tâche aux cibles de tâche spécifiées sur lesquelles un agent de tâches applique le document de tâche à l'appareil ou au client.

Procédures décrites dans cette section

- Créez et stockez le document de travail pour la tâche IoT
- Exécuter une tâche AWS IoT pour un appareil IoT

Créez et stockez le document de travail pour la tâche IoT

Cette procédure crée un document de travail simple à inclure dans une ressource de AWS IoT travail. Ce document job affiche « Hello world ! » sur l'objectif du job.

Pour créer et stocker un document job, procédez comme suit :

1. Sélectionnez le compartiment Amazon S3 dans lequel vous allez enregistrer votre document job. Si vous n'avez pas de compartiment Amazon S3 à utiliser à cette fin, vous aurez besoin d'en créer. Pour plus d'informations sur la création de compartiments Amazon S3, consultez les rubriques Démarrage avec Amazon S3.
2. Créer et enregistrer le document job pour cette tâche

   a. Sur votre ordinateur hôte local, ouvrez un éditeur de texte.

   b. Copiez et collez ce texte dans l'éditeur.

```
{
    "operation": "echo",
    "args": ["Hello world!"]
}
```

c. Sur l'ordinateur hôte local, enregistrez le contenu de l'éditeur dans un fichier nommé **`hello-world-job.json`**.

d. Vérifiez que le fichier a été correctement enregistré. Certains éditeurs de texte ajoutent automatiquement `.txt` au nom du fichier lorsqu'ils enregistrent un fichier texte. Si votre éditeur a ajouté `.txt` au nom du fichier, corrigez-le avant de continuer.

3. Remplacez le *`path_to_file`* par le chemin vers**`hello-world-job.json`**, s'il ne figure pas dans votre répertoire actuel, remplacez-le *`s3_bucket_name`* par le chemin du compartiment Amazon S3 vers le compartiment que vous avez sélectionné, puis exécutez cette commande pour placer votre document de travail dans le compartiment Amazon S3.

```
aws s3api put-object \
--key hello-world-job.json \
--body path_to_file/hello-world-job.json --bucket s3_bucket_name
```

L'URL du document de travail qui identifie le document de travail que vous avez stocké dans Amazon S3 est déterminée en remplaçant le *`s3_bucket_name`* et *`AWS_region`* dans l'URL suivante. Enregistrez l'URL résultante pour l'utiliser ultérieurement en tant que *`job_document_path`*

```
https://s3_bucket_name.s3.AWS_Region.amazonaws.com/hello-world-job.json
```

> **Note**
>
> AWS la sécurité vous empêche d'ouvrir cette URL en dehors de la vôtre Compte AWS, par exemple à l'aide d'un navigateur. L'URL est utilisée par le moteur de AWS IoT tâches, qui a accès au fichier par défaut. Dans un environnement de production, vous devez vous assurer que vos services AWS IoT sont autorisés à accéder aux documents job stockés dans Amazon S3.

Après avoir enregistré l'URL du document job, passez à the section called “Exécuter une tâche pour un seul appareil”.

Exécuter une tâche AWS IoT pour un appareil IoT

Les procédures décrites dans cette section démarrent le client de AWS IoT périphérique sur votre Raspberry Pi pour exécuter l'agent de tâches sur le périphérique afin d'attendre l'exécution des

tâches. Cela crée également une ressource de travail dans AWS IoT, qui enverra la tâche et s'exécutera sur votre appareil IoT.

> **Note**
>
> Cette procédure exécute une job sur un seul appareil.

Pour démarrer l'agent de jobs sur votre Raspberry Pi :

1. Dans la fenêtre du terminal de votre ordinateur hôte local connecté à votre Raspberry Pi, exécutez cette commande pour démarrer le AWS IoT Device Client.

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-jobs-config.json
```

2. Dans la fenêtre du terminal, vérifiez que le client du AWS IoT périphérique affiche ces messages

```
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Jobs is enabled
                        .
                        .
                        .
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Client base has been notified that
 Jobs has started
2021-11-15T18:45:56.708Z [INFO]  {JobsFeature.cpp}: Running Jobs!
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 startNextPendingJobExecution accepted and rejected
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 nextJobChanged events
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusAccepted for jobId +
2021-11-15T18:45:56.738Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionAccepted with code {0}
2021-11-15T18:45:56.739Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusRejected for jobId +
2021-11-15T18:45:56.753Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToNextJobChanged with code {0}
2021-11-15T18:45:56.760Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobRejected with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobAccepted with code {0}
```

```
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionRejected with code {0}
2021-11-15T18:45:56.777Z [DEBUG] {JobsFeature.cpp}: Publishing
 startNextPendingJobExecutionRequest
2021-11-15T18:45:56.785Z [DEBUG] {JobsFeature.cpp}: Ack received for
 StartNextPendingJobPub with code {0}
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. Dans la fenêtre du terminal, après avoir vu ce message, passez à la procédure suivante et créez la ressource job. Notez qu'il ne s'agit peut-être pas de la dernière entrée de la liste.

```
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

Pour créer une ressource AWS IoT d'emploi

1. Sur votre ordinateur hôte local :

   a. Remplacez *job_document_url* par l'URL du document de travail provenant dethe section called “Création et stockage d'un document de travail”.

   b. *thing_arn*Remplacez-le par l'ARN de la ressource d'objet que vous avez créée pour votre appareil, puis exécutez cette commande.

   ```
   aws iot create-job \
   --job-id hello-world-job-1 \
   --document-source "job_document_url" \
   --targets "thing_arn" \
   --target-selection SNAPSHOT
   ```

   En cas de succès, la commande renvoie un résultat comme celui-ci.

   ```
   {
     "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",
     "jobId": "hello-world-job-1"
   }
   ```

2. Dans la fenêtre du terminal, vous devriez voir une sortie du client de AWS IoT périphérique comme celle-ci.

```
2021-11-15T18:02:26.688Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Job ids differ
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: Executing job: hello-world-
job-1
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Assuming executable is in PATH
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: About to execute: echo Hello
 world!
2021-11-15T18:10:24.890Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken 3TEWba9Xj6 in the updateJobExecution promises map
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process now running
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process about to call
 execvp
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Parent process now running, child
 PID is 16737
2021-11-15T18:10:24.891Z [DEBUG] {16737}: Hello world!
2021-11-15T18:10:24.891Z [DEBUG] {JobEngine.cpp}: JobEngine finished waiting for
 child process, returning 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job exited with status: 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job executed successfully!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Attempting to update job
 execution status!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.892Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.892Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken GmQ0HTzWGg in the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken 3TEWba9Xj6
 from the updateJobExecution promises map
```

```
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:24.917Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken GmQ0HTzWGg
 from the updateJobExecution promises map
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:25.861Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. Pendant que le AWS IoT Device Client est en cours d'exécution et attend une tâche, vous pouvez en soumettre une autre en modifiant la `job-id` valeur et en la réexécutant create-job depuis l'étape 1.

Lorsque vous avez terminé d'exécuter les tâches, dans la fenêtre du terminal, entrez ^C (Control-C) pour arrêter le AWS IoT Device Client.

Tutoriel : Nettoyage après l'exécution des didacticiels AWS IoT Device Client

Les procédures décrites dans ce didacticiel vous expliquent comment supprimer les fichiers et les ressources que vous avez créés tout en suivant les didacticiels de ce parcours d'apprentissage.

Procédures décrites dans ce didacticiel

- Étape 1 : Nettoyage de vos appareils après avoir créé des démos avec le AWS IoT Device Client
- Étape 2 : Nettoyez vos démos Compte AWS après avoir créé des démos avec le AWS IoT Device Client

Étape 1 : Nettoyage de vos appareils après avoir créé des démos avec le AWS IoT Device Client

Ce didacticiel décrit deux options permettant de nettoyer la carte microSD après avoir créé les démos de ce parcours d'apprentissage. Choisissez l'option qui fournit le niveau de sécurité dont vous avez besoin.

Notez que le nettoyage de la carte microSD de l'appareil ne supprime aucune AWS IoT ressource que vous avez créée. Pour nettoyer les AWS IoT ressources après avoir nettoyé la carte microSD de

l'appareil, consultez le didacticiel surthe section called “Nettoyage après la création de démos avec le AWS IoT Device Client”.

Option 1 : Nettoyage en réécrivant la carte microSD

La méthode la plus simple et la plus complète pour nettoyer la carte microSD après avoir suivi les didacticiels de ce parcours d'apprentissage consiste à remplacer la carte microSD par un fichier image enregistré que vous avez créé lors de la première préparation de votre appareil.

Cette procédure utilise l'ordinateur hôte local pour écrire une image de carte microSD enregistrée sur une carte microSD.

> **Note**
>
> Si votre appareil n'utilise pas de support de stockage amovible pour son système d'exploitation, reportez-vous à la procédure correspondant à cet appareil.

Pour écrire une nouvelle image sur la carte microSD

1. Sur votre ordinateur hôte local, localisez l'image de la carte microSD enregistrée que vous souhaitez écrire sur votre carte microSD.
2. Insérez votre carte microSD dans l'ordinateur hôte local.
3. À l'aide d'un outil d'imagerie sur carte SD, écrivez le fichier image sélectionné sur la carte microSD.
4. Après avoir enregistré l'image du système d'exploitation du Raspberry Pi sur la carte microSD, éjectez la carte microSD et retirez-la en toute sécurité de l'ordinateur hôte local.

Votre carte microSD est prête à être utilisée.

Option 2 : Nettoyage en supprimant les annuaires des utilisateurs

Pour nettoyer la carte microSD après avoir terminé les didacticiels sans réécrire l'image de la carte microSD, vous pouvez supprimer les répertoires utilisateur individuellement. Cela n'est pas aussi complet que de réécrire la carte microSD à partir d'une image enregistrée, car cela ne supprime aucun fichier système qui aurait pu être installé.

Si la suppression des annuaires d'utilisateurs est suffisamment complète pour répondre à vos besoins, vous pouvez suivre cette procédure.

Pour supprimer les répertoires des utilisateurs de ce parcours de formation sur votre appareil

1. Exécutez ces commandes pour supprimer les répertoires utilisateur, les sous-répertoires et tous leurs fichiers qui ont été créés dans le cadre de ce parcours d'apprentissage, dans la fenêtre du terminal connectée à votre appareil.

   > **Note**
   >
   > Après avoir supprimé ces répertoires et fichiers, vous ne pourrez plus exécuter les démos sans avoir terminé à nouveau les didacticiels.

   ```
   rm -Rf ~/dc-configs
   rm -Rf ~/policies
   rm -Rf ~/messages
   rm -Rf ~/certs
   rm -Rf ~/.aws-iot-device-client
   ```

2. Exécutez ces commandes pour supprimer les répertoires et les fichiers source de l'application dans la fenêtre du terminal connectée à votre appareil.

   > **Note**
   >
   > Ces commandes ne désinstallent aucun programme. Ils suppriment uniquement les fichiers source utilisés pour les créer et les installer. Une fois ces fichiers supprimés, le AWS CLI et le client de l'AWS IoTappareil peuvent ne pas fonctionner.

   ```
   rm -Rf ~/aws-cli
   rm -Rf ~/aws
   rm -Rf ~/aws-iot-device-client
   ```

Étape 2 : Nettoyez vos démos Compte AWS après avoir créé des démos avec le AWS IoT Device Client

Ces procédures vous aident à identifier et à supprimer les AWS ressources que vous avez créées en suivant les didacticiels de ce parcours de formation.

Nettoyez les AWS IoT ressources

Cette procédure vous permet d'identifier et de supprimer les AWS IoT ressources que vous avez créées en suivant les didacticiels de ce parcours de formation.

AWS IoTressources créées dans ce parcours d'apprentissage

Didacticiel	Ressource d'objets	Ressource de politique
the section called “Installation et configuration d'un client pour appareils IoT”	DevCliTestThing	DevCliTestThingPolicy
the section called “Communiquez avec le client Device à l'aide de MQTT”	PubSubTestThing	PubSubTestThingPolicy
the section called “Exécutez des tâches IoT avec le Device Client”	défini par l'utilisateur (il peut y en avoir plusieurs)	défini par l'utilisateur (il peut y en avoir plusieurs)

Pour supprimer les AWS IoT ressources, suivez cette procédure pour chaque ressource que vous avez créée

1. Remplacez-le *thing_name* par le nom de la ressource objet que vous souhaitez supprimer, puis exécutez cette commande pour répertorier les certificats attachés à la ressource objet, à partir de l'ordinateur hôte local.

   ```
   aws iot list-thing-principals --thing-name thing_name
   ```

   Cette commande renvoie une réponse comme celle-ci qui répertorie les certificats attachés à*thing_name*. Dans la plupart des cas, il n'y aura qu'un seul certificat dans la liste.

   ```
   {
       "principals": [
           "arn:aws:iot:us-
   west-2:57EXAMPLE833:cert/23853eea3cf0edc7f8a69c74abeafa27b2b52823cab5b3e156295e94b26ae8ac"
       ]
   }
   ```

2. Pour chaque certificat répertorié par la commande précédente :

   a. *certificate_ID*Remplacez-le par l'ID du certificat de la commande précédente. L'ID du certificat est constitué des caractères alphanumériques qui suivent `cert/` dans l'ARN renvoyé par la commande précédente. Exécutez ensuite cette commande pour désactiver le certificat.

   ```
   aws iot update-certificate --new-status INACTIVE --certificate-
   id certificate_ID
   ```

   En cas de succès, cette commande ne renvoie rien.

   b. Remplacez-le *certificate_ARN* par l'ARN du certificat dans la liste des certificats renvoyée précédemment, puis exécutez cette commande pour répertorier les politiques associées à ce certificat.

   ```
   aws iot list-attached-policies --target certificate_ARN
   ```

   Cette commande renvoie une réponse comme celle-ci qui répertorie les politiques associées au certificat. Dans la plupart des cas, il n'y aura qu'une seule politique dans la liste.

   ```
   {
       "policies": [
           {
               "policyName": "DevCliTestThingPolicy",
               "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
   DevCliTestThingPolicy"
           }
       ]
   }
   ```

   c. Pour chaque politique associée au certificat :

      i. *policy_name*Remplacez-la par la `policyName` valeur de la commande précédente, *certificate_ARN* remplacez-la par l'ARN du certificat, puis exécutez cette commande pour détacher la politique du certificat.

      ```
      aws iot detach-policy --policy-name policy_name --target certificate_ARN
      ```

      En cas de succès, cette commande ne renvoie rien.

ii. *policy_name*Remplacez-le par la `policyName` valeur, puis exécutez cette commande pour voir si la politique est associée à d'autres certificats.

```
aws iot list-targets-for-policy --policy-name policy_name
```

Si la commande renvoie une liste vide comme celle-ci, la politique n'est associée à aucun certificat et vous continuez à répertorier les versions de la politique. Si des certificats sont toujours associés à la politique, passez à l'detach-thing-principalétape suivante.

```
{
    "targets": []
}
```

iii. *policy_name*Remplacez-le par la `policyName` valeur, puis exécutez cette commande pour vérifier les versions des politiques. Pour supprimer la politique, il ne doit y avoir qu'une seule version.

```
aws iot list-policy-versions --policy-name policy_name
```

Si la politique ne comporte qu'une seule version, comme dans cet exemple, vous pouvez passer à l'delete-policyétape suivante et supprimer la politique dès maintenant.

```
{
    "policyVersions": [
        {
            "versionId": "1",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:02:46.778000+00:00"
        }
    ]
}
```

Si la politique comporte plusieurs versions, comme dans cet exemple, les versions de stratégie dont la `isDefaultVersion` valeur est `false` doivent être supprimées avant que la politique puisse être supprimée.

```
{
    "policyVersions": [
```

```
        {
            "versionId": "2",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:52:04.423000+00:00"
        },
        {
            "versionId": "1",
            "isDefaultVersion": false,
            "createDate": "2021-11-18T01:30:18.083000+00:00"
        }
    ]
}
```

Si vous devez supprimer une version de politique, remplacez-la *policy_name* par la `policyName` valeur, *version_ID* remplacez-la par la `versionId` valeur de la commande précédente, puis exécutez cette commande pour supprimer une version de stratégie.

```
aws iot delete-policy-version --policy-name policy_name --policy-version-
id version_ID
```

En cas de succès, cette commande ne renvoie rien.

Après avoir supprimé une version de politique, répétez cette étape jusqu'à ce que la politique ne comporte qu'une seule version de stratégie.

iv. *policy_name*Remplacez-le par la `policyName` valeur, puis exécutez cette commande pour supprimer la politique.

```
aws iot delete-policy --policy-name policy_name
```

d. Remplacez *thing_name* par le nom de l'objet, remplacez-le *certificate_ARN* par l'ARN du certificat, puis exécutez cette commande pour détacher le certificat de la ressource de l'objet.

```
aws iot detach-thing-principal --thing-name thing_name --
principal certificate_ARN
```

En cas de succès, cette commande ne renvoie rien.

e. *certificate_ID*Remplacez-le par l'ID du certificat de la commande précédente. L'ID du certificat est constitué des caractères alphanumériques qui suivent `cert/` dans l'ARN renvoyé par la commande précédente. Exécutez ensuite cette commande pour supprimer la ressource de certificat.

```
aws iot delete-certificate --certificate-id certificate_ID
```

En cas de succès, cette commande ne renvoie rien.

3. Remplacez *thing_name* par le nom de l'objet, puis exécutez cette commande pour supprimer l'objet.

```
aws iot delete-thing --thing-name thing_name
```

En cas de succès, cette commande ne renvoie rien.

Nettoyez les AWS ressources

Cette procédure vous permet d'identifier et de supprimer les autres AWS ressources que vous avez créées en suivant les didacticiels de ce parcours de formation.

Autres AWS ressources créées dans le cadre de ce parcours d'apprentissage

Didacticiel	Type de ressource	Nom ou ID de la ressource
the section called “Exécutez des tâches IoT avec le Device Client”	Objet Amazon S3	hello-world-job.json
the section called “Exécutez des tâches IoT avec le Device Client”	AWS IoTressources d'emploi	défini par l'utilisateur

Pour supprimer les AWS ressources créées dans ce parcours de formation

1. Pour supprimer les emplois créés dans ce parcours de formation

a. Exécutez cette commande pour répertorier les tâches de votreCompte AWS.

```
aws iot list-jobs
```

La commande renvoie une liste des AWS IoT tâches de votre Compte AWS et Région AWS qui ressemble à ceci.

```
{
    "jobs": [
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
job-2",
            "jobId": "hello-world-job-2",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:40:36.825000+00:00",
            "lastUpdatedAt": "2021-11-16T23:40:41.375000+00:00",
            "completedAt": "2021-11-16T23:40:41.375000+00:00"
        },
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-
job-1",
            "jobId": "hello-world-job-1",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:35:26.381000+00:00",
            "lastUpdatedAt": "2021-11-16T23:35:29.239000+00:00",
            "completedAt": "2021-11-16T23:35:29.239000+00:00"
        }
    ]
}
```

b. Pour chaque tâche que vous reconnaissez dans la liste comme une tâche que vous avez créée dans ce parcours de formation, *`jobId`* remplacez-la par la `jobId` valeur de la tâche à supprimer, puis exécutez cette commande pour supprimer une AWS IoT tâche.

```
aws iot delete-job --job-id jobId
```

Si la commande aboutit, elle ne renvoie rien.

2. Pour supprimer les documents de travail que vous avez stockés dans un compartiment Amazon S3 dans ce parcours de formation.

a. Remplacez-le *bucket* par le nom du compartiment que vous avez utilisé, puis exécutez cette commande pour répertorier les objets du compartiment Amazon S3 que vous avez utilisé.

```
aws s3api list-objects --bucket bucket
```

La commande renvoie une liste des objets Amazon S3 contenus dans le compartiment qui ressemble à ceci.

```
{
    "Contents": [
        {
            "Key": "hello-world-job.json",
            "LastModified": "2021-11-18T03:02:12+00:00",
            "ETag": "\"868c8bc3f56b5787964764d4b18ed5ef\"",
            "Size": 54,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "iot_job_firmware_update.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"7c68c591949391791ecf625253658c61\"",
            "Size": 66,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "order66.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"bca60d5380b88e1a70cc27d321caba72\"",
            "Size": 29,
            "StorageClass": "STANDARD",
```

```
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        }
    ]
}
```

b. Pour chaque objet que vous reconnaissez dans la liste comme étant un objet que vous avez créé dans ce parcours d'apprentissage, remplacez-le par *bucket* le nom du compartiment et *key* par la valeur clé de l'objet à supprimer, puis exécutez cette commande pour supprimer un objet Amazon S3.

```
aws s3api delete-object --bucket bucket --key key
```

Si la commande aboutit, elle ne renvoie rien.

Après avoir supprimé toutes les AWS ressources et tous les objets que vous avez créés au cours de ce parcours d'apprentissage, vous pouvez recommencer à zéro et reprendre les didacticiels.

Création de solutions à l'aide de l' AWS IoT appareil SDKs

Les didacticiels de cette section vous aident à suivre les étapes du développement d'une solution IoT pouvant être déployée dans un environnement de production à l'aide de AWS IoT.

Ces didacticiels peuvent prendre plus de temps que ceux présentés dans la section correspondante, the section called “Création de démos avec le AWS IoT Device Client” car ils utilisent l' AWS IoT appareil SDKs et expliquent les concepts appliqués de manière plus détaillée pour vous aider à créer des solutions sécurisées et fiables.

Commencez à créer des solutions avec l' AWS IoT appareil SDKs

Ces didacticiels vous présentent différents AWS IoT scénarios. Le cas échéant, les didacticiels utilisent l' AWS IoT appareil SDKs.

Rubriques

- Tutoriel : Connexion d'un appareil à AWS IoT Core l'aide de l' AWS IoT appareil SDK
- Création de AWS IoT règles pour acheminer les données des appareils vers d'autres services

- Conservation de l'état de l'appareil lorsque celui-ci est hors ligne avec Device Shadows
- Didacticiel : Création d'un mécanisme d'autorisation personnalisé pour AWS IoT Core
- Tutoriel : Surveillance de l'humidité du sol avec un AWS IoT Raspberry Pi

Tutoriel : Connexion d'un appareil à AWS IoT Core l'aide de l' AWS IoT appareil SDK

Ce didacticiel explique comment connecter un appareil AWS IoT Core afin qu'il puisse envoyer et recevoir des données depuis et vers AWS IoT. Une fois ce didacticiel terminé, votre appareil sera configuré pour se connecter à AWS IoT Core et vous comprendrez comment les appareils communiquent avec AWS IoT.

Rubriques

- Prérequis
- Préparez votre appareil pour AWS IoT
- Passez en revue le MQTT protocole
- Consultez l'SDKexemple d'application pubsub.py pour appareil
- Connectez votre appareil et communiquez avec AWS IoT Core
- Passez en revue les résultats
- Tutoriel : Utilisation du Kit SDK des appareils AWS IoT pour Embedded C

Prérequis

Avant de commencer ce didacticiel, assurez-vous de disposer des éléments suivants :

- Terminé Commencer à utiliser les AWS IoT Core didacticiels

  Dans la section de ce didacticiel où vous devezthe section called “Configurer votre appareil”, sélectionnez l'the section called “Connectez un Raspberry Pi ou un autre appareil”option correspondant à votre appareil et utilisez les options du langage Python pour configurer votre appareil.

> **Note**
>
> Gardez ouverte la fenêtre du terminal que vous utilisez dans ce didacticiel, car vous l'utiliserez également dans ce didacticiel.

- Un appareil capable d'exécuter le AWS IoT Device SDK v2 pour Python.

  Ce didacticiel montre comment connecter un appareil à l'aide AWS IoT Core d'exemples de code Python, qui nécessitent un périphérique relativement puissant. Si vous travaillez avec des appareils dont les ressources sont limitées, il est possible que ces exemples de code ne fonctionnent pas sur eux. Dans ce cas, vous aurez peut-être plus de succès avec le the section called “En utilisant le Kit SDK des appareils AWS IoT pour Embedded C” didacticiel.

- Vous avez obtenu les informations requises pour vous connecter à l'appareil

  Pour connecter votre appareil à AWS IoT, vous devez disposer d'informations sur le nom de l'objet, le nom d'hôte et le numéro de port.

  > **Note**
  >
  > Vous pouvez également utiliser l'authentification personnalisée pour connecter des appareils à AWS IoT Core. Les données de connexion que vous transmettez à votre fonction Lambda d'autorisation dépendent du protocole que vous utilisez.

  - Nom de l'objet : nom de l' AWS IoT objet auquel vous souhaitez vous connecter. Vous devez avoir enregistré votre appareil en tant qu' AWS IoT objet. Pour de plus amples informations, veuillez consulter Gestion des appareils avec AWS IoT.
  - Nom d'hôte : nom d'hôte du point de terminaison IoT spécifique au compte.
  - Numéro de port : numéro de port auquel se connecter.

  Vous pouvez utiliser la `configureEndpoint` méthode en AWS IoT Python SDK pour configurer le nom d'hôte et le numéro de port.

  ```
  myAWSIoTMQTTClient.configureEndpoint("random.iot.region.amazonaws.com", 8883)
  ```

Préparez votre appareil pour AWS IoT

Dans Commencer à utiliser les AWS IoT Core didacticiels, vous avez préparé votre appareil et votre AWS compte pour qu'ils puissent communiquer. Cette section passe en revue les aspects de cette préparation qui s'appliquent à toute connexion d'appareil avec AWS IoT Core.

Pour qu'un appareil puisse se connecter à AWS IoT Core :

1. Vous devez avoir un Compte AWS.

   La procédure Configurez Compte AWS décrite dans décrit comment créer un Compte AWS si vous n'en avez pas déjà un.
2. Dans ce compte, les AWS IoT ressources suivantes doivent être définies pour l'appareil de votre région Compte AWS et de votre région.

   La procédure décrite ci-dessous Créez des AWS IoT ressources décrit comment créer ces ressources pour l'appareil dans votre région Compte AWS .
   - Un certificat d'appareil enregistré AWS IoT et activé pour authentifier l'appareil.

     Le certificat est souvent créé avec un AWS IoT objet et y est attaché. Bien qu'il ne soit pas nécessaire qu'un appareil se connecte à un objet AWS IoT, il met des AWS IoT fonctionnalités supplémentaires à la disposition de celui-ci.
   - Politique attachée au certificat de l'appareil qui l'autorise à se connecter AWS IoT Core et à effectuer toutes les actions que vous souhaitez qu'il effectue.
3. Une connexion Internet qui peut accéder aux terminaux Compte AWS de votre appareil.

   Les points de terminaison de l'appareil sont décrits AWS IoT données de l'appareil et points de terminaison de service et peuvent être consultés sur la page des paramètres de la AWS IoT console.
4. Logiciel de communication tel que celui SDKs fourni par l' AWS IoT appareil. Ce tutoriel utilise le AWS IoT Device SDK v2 pour Python.

Passez en revue le MQTT protocole

Avant de parler de l'exemple d'application, il est utile de comprendre le MQTT protocole. Le MQTT protocole offre certains avantages par rapport aux autres protocoles de communication réseau, notammentHTTP, ce qui en fait un choix populaire pour les appareils IoT. Cette section passe en revue les principaux aspects MQTT qui s'appliquent à ce didacticiel. Pour plus d'informations

sur le MQTT mode de comparaison avecHTTP, voirChoix d'un protocole d'application pour la communication de votre appareil.

MQTTutilise un modèle de communication publication/abonnement

Le MQTT protocole utilise un publish/subscribe communication model with its host. This model differs from the request/response modèle qui HTTP utilise. AvecMQTT, les appareils établissent une session avec l'hôte qui est identifié par un identifiant client unique. Pour envoyer des données, les appareils publient des messages identifiés par sujets à un courtier de messages sur l'hôte. Pour recevoir des messages du courtier de messages, les appareils s'abonnent aux sujets en envoyant des filtres de sujets dans les demandes d'abonnement adressées au courtier de messages.

MQTTprend en charge les sessions persistantes

Le courtier de messages reçoit les messages des appareils et publie des messages aux appareils qui y sont abonnés. Avec les sessions persistantes, —c'est-à-dire des sessions qui restent actives même lorsque l'appareil initiateur est déconnecté— les appareils peuvent récupérer les messages publiés alors qu'ils étaient déconnectés. Du côté de l'appareil, MQTT prend en charge les niveaux de qualité de service (QoS) qui garantissent que l'hôte reçoit les messages envoyés par l'appareil.

Consultez l'SDKexemple d'application pubsub.py pour appareil

Cette section passe en revue l'`pubsub.py`exemple d'application du AWS IoT Device SDK v2 pour Python utilisé dans ce didacticiel. Ici, nous allons voir comment il se connecte AWS IoT Core pour publier des MQTT messages et s'y abonner. La section suivante présente quelques exercices destinés à vous aider à découvrir comment un appareil se connecte et communique avec lui AWS IoT Core.

L'pubsub.pyexemple d'application illustre les aspects suivants d'une MQTT connexion avec AWS IoT Core :

- Protocoles de communication
- Sessions persistantes
- Qualité du service
- Publication du message
- Abonnement aux messages
- Déconnexion et reconnexion de l'appareil

Protocoles de communication

L'`pubsub.py`exemple montre une MQTT connexion utilisant les WSS protocoles MQTT and MQTT over. La bibliothèque AWS common runtime (AWS CRT) fournit le support des protocoles de communication de bas niveau et est incluse dans le AWS IoT Device SDK v2 pour Python.

MQTT

Les `pubsub.py` exemples d'appels `mtls_from_path` (présentés ici) dans le `mqtt_connection_builder`pour établir une connexion à AWS IoT Core l'aide du MQTT protocole. `mtls_from_path`utilise les certificats X.509 et TLS v1.2 pour authentifier le périphérique. La AWS CRT bibliothèque gère les détails de niveau inférieur de cette connexion.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
    endpoint=args.endpoint,
    cert_filepath=args.cert,
    pri_key_filepath=args.key,
    ca_filepath=args.ca_file,
    client_bootstrap=client_bootstrap,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

Le point Compte AWS de terminaison de votre appareil IoT

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`cert_filepath`

Chemin d'accès au fichier de certificat de l'appareil

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`pri_key_filepath`

Le chemin d'accès au fichier de clé privée de l'appareil créé avec son fichier de certificat

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`ca_filepath`

Chemin d'accès au fichier de l'autorité de certification racine. Obligatoire uniquement si le MQTT serveur utilise un certificat qui ne figure pas déjà dans votre trust store.

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`client_bootstrap`

L'objet d'exécution commun qui gère les activités de communication avec les sockets

Dans l'exemple d'application, cet objet est instancié avant l'appel à `mqtt_connection_builder.mtls_from_path`.

`on_connection_interrupted`, `on_connection_resumed`

Les fonctions de rappel permettent d'appeler lorsque la connexion de l'appareil est interrompue et reprise

`client_id`

L'identifiant qui identifie de manière unique cet appareil dans Région AWS

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`clean_session`

S'il faut démarrer une nouvelle session persistante ou, le cas échéant, se reconnecter à une session existante

`keep_alive_secs`

La valeur de maintien en vie, en secondes, à envoyer à la `CONNECT` demande. Un ping sera automatiquement envoyé à cet intervalle. Si le serveur ne reçoit pas de ping après 1,5 fois cette valeur, il suppose que la connexion est perdue.

MQTTterminé WSS

Les `pubsub.py` exemples d'appels `websockets_with_default_aws_signing` (présentés ici) dans le `mqtt_connection_builder`pour établir une connexion en AWS IoT Core utilisant le MQTT protocole overWSS. `websockets_with_default_aws_signing`crée une MQTT connexion à l'WSSaide de Signature V4 pour authentifier l'appareil.

```
mqtt_connection = mqtt_connection_builder.websockets_with_default_aws_signing(
```

```
    endpoint=args.endpoint,
    client_bootstrap=client_bootstrap,
    region=args.signing_region,
    credentials_provider=credentials_provider,
    websocket_proxy_options=proxy_options,
    ca_filepath=args.ca_file,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

Le point Compte AWS de terminaison de votre appareil IoT

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`client_bootstrap`

L'objet d'exécution commun qui gère les activités de communication avec les sockets

Dans l'exemple d'application, cet objet est instancié avant l'appel à `mqtt_connection_builder.websockets_with_default_aws_signing`

`region`

Région de AWS signature utilisée par l'authentification Signature V4. Dans `pubsub.py`, il transmet le paramètre saisi dans la ligne de commande.

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`credentials_provider`

Les AWS informations d'identification fournies à utiliser pour l'authentification

Dans l'exemple d'application, cet objet est instancié avant l'appel à `mqtt_connection_builder.websockets_with_default_aws_signing`.

`websocket_proxy_options`

HTTPoptions de proxy, si vous utilisez un hôte proxy

Dans l'exemple d'application, cette valeur est initialisée avant l'appel à `mqtt_connection_builder.websockets_with_default_aws_signing`.

`ca_filepath`

Chemin d'accès au fichier de l'autorité de certification racine. Obligatoire uniquement si le MQTT serveur utilise un certificat qui ne figure pas déjà dans votre trust store.

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`on_connection_interrupted`, `on_connection_resumed`

Les fonctions de rappel permettent d'appeler lorsque la connexion de l'appareil est interrompue et reprise

`client_id`

L'identifiant qui identifie de manière unique cet appareil dans le Région AWS.

Dans l'exemple d'application, cette valeur est transmise depuis la ligne de commande.

`clean_session`

S'il faut démarrer une nouvelle session persistante ou, le cas échéant, se reconnecter à une session existante

`keep_alive_secs`

La valeur de maintien en vie, en secondes, à envoyer à la `CONNECT` demande. Un ping sera automatiquement envoyé à cet intervalle. Si le serveur ne reçoit pas de ping après 1,5 fois cette valeur, il suppose que la connexion est perdue.

HTTPS

Qu'en est-il HTTPS ? AWS IoT Core prend en charge les appareils qui publient HTTPS des demandes. Du point de vue de la programmation, les appareils envoient HTTPS des demandes AWS IoT Core comme le ferait n'importe quelle autre application. Pour un exemple de programme Python qui envoie un HTTP message depuis un appareil, consultez l'exemple de HTTPS code utilisant la `requests` bibliothèque Python. Cet exemple envoie un message à AWS IoT Core un HTTPS utilisateur qui l' AWS IoT Core interprète comme un MQTT message.

Tout AWS IoT Core en prenant en charge les HTTPS demandes des appareils, assurez-vous de consulter les informations Choix d'un protocole d'application pour la communication de votre appareil correspondantes afin de pouvoir prendre une décision éclairée quant au protocole à utiliser pour les communications de votre appareil.

Sessions persistantes

Dans l'exemple d'application, la définition du `clean_session` paramètre sur `False` indique que la connexion doit être permanente. En pratique, cela signifie que la connexion ouverte par cet appel se reconnecte à une session persistante existante, s'il en existe une. Dans le cas contraire, il crée une nouvelle session persistante et s'y connecte.

Dans le cas d'une session permanente, les messages envoyés à l'appareil sont stockés par le courtier de messages lorsque l'appareil n'est pas connecté. Lorsqu'un appareil se reconnecte à une session permanente, le courtier de messages envoie au terminal tous les messages enregistrés auxquels il est abonné.

Sans session permanente, l'appareil ne recevra pas les messages envoyés lorsqu'il n'est pas connecté. L'option à utiliser dépend de votre application et de la nécessité de communiquer les messages qui apparaissent alors qu'un appareil n'est pas connecté. Pour de plus amples informations, veuillez consulter Sessions permanentes MQTT.

Qualité du service

Lorsque l'appareil publie des messages et s'y abonne, la qualité de service (QoS) préférée peut être définie. AWS IoT prend en charge les niveaux de QoS 0 et 1 pour les opérations de publication et d'abonnement. Pour plus d'informations sur les niveaux de QoS dans AWS IoT, consultez. Options de qualité de service (QoS) MQTT

Le AWS CRT moteur d'exécution de Python définit les constantes suivantes pour les niveaux de QoS qu'il prend en charge :

Niveaux de qualité de service en Python

MQTTNiveau de QoS	Valeur symbolique Python utilisée par SDK	Description
QoS niveau 0	`mqtt.QoS.AT_MOST_ONCE`	Une seule tentative d'envoi du message sera effectuée, qu'il soit reçu ou non. Le message peut ne pas être envoyé du tout, par exemple si l'appareil n'est pas connecté ou s'il y a une erreur réseau.

MQTTNiveau de QoS	Valeur symbolique Python utilisée par SDK	Description
QoS niveau 1	`mqtt.QoS.AT_LEAST_ONCE`	Le message est envoyé à plusieurs reprises jusqu'à ce qu'un PUBACK accusé de réception soit reçu.

Dans l'exemple d'application, les demandes de publication et d'abonnement sont effectuées avec un niveau de QoS de 1 (`mqtt.QoS.AT_LEAST_ONCE`).

- QoS lors de la publication

  Lorsqu'un appareil publie un message avec le niveau de QoS 1, il l'envoie à plusieurs reprises jusqu'à ce qu'il reçoive une PUBACK réponse du courtier de messages. Si l'appareil n'est pas connecté, le message est mis en file d'attente pour être envoyé après sa reconnexion.
- QoS lors de l'abonnement

  Lorsqu'un appareil s'abonne à un message avec QoS de niveau 1, le courtier de messages enregistre les messages auxquels le périphérique est abonné jusqu'à ce qu'ils puissent être envoyés au périphérique. Le courtier de messages renvoie les messages jusqu'à ce qu'il reçoive une PUBACK réponse de l'appareil.

Publication du message

Une fois la connexion établie avec succès AWS IoT Core, les appareils peuvent publier des messages. Pour ce faire, l'`pubsub.py`exemple appelle le `publish` fonctionnement de l'`mqtt_connection`objet.

```
mqtt_connection.publish(
    topic=args.topic,
    payload=message,
    qos=mqtt.QoS.AT_LEAST_ONCE
)
```

`topic`

Le nom du sujet du message qui identifie le message

Dans l'exemple d'application, cela est transmis depuis la ligne de commande.

`payload`

La charge utile du message formatée sous forme de chaîne (par exemple, un JSON document)

Dans l'exemple d'application, cela est transmis depuis la ligne de commande.

Un JSON document est un format de charge utile courant, reconnu par d'autres AWS IoT services ; toutefois, le format de données de la charge utile du message peut être celui sur lequel les éditeurs et les abonnés sont d'accord. Cependant, d'autres AWS IoT services reconnaissent uniquement et JSONCBOR, dans certains cas, pour la plupart des opérations.

`qos`

Le niveau de QoS pour ce message

Abonnement aux messages

Pour recevoir des messages provenant AWS IoT d'autres services et appareils, les appareils s'abonnent à ces messages en utilisant le nom de leur sujet. Les appareils peuvent s'abonner à des messages individuels en spécifiant un nom de sujet, et à un groupe de messages en spécifiant un filtre de sujet, qui peut inclure des caractères génériques. L'`pubsub.py`exemple utilise le code présenté ici pour s'abonner aux messages et enregistrer les fonctions de rappel afin de traiter le message après sa réception.

```
subscribe_future, packet_id = mqtt_connection.subscribe(
    topic=args.topic,
    qos=mqtt.QoS.AT_LEAST_ONCE,
    callback=on_message_received
)
subscribe_result = subscribe_future.result()
```

`topic`

Le sujet auquel s'abonner. Il peut s'agir d'un nom de rubrique ou d'un filtre de rubrique.

Dans l'exemple d'application, cela est transmis depuis la ligne de commande.

`qos`

Si le courtier de messages doit stocker ces messages lorsque l'appareil est déconnecté.

Une valeur de `mqtt.QoS.AT_LEAST_ONCE` (QoS niveau 1) nécessite qu'une session persistante soit spécifiée (`clean_session=False`) lors de la création de la connexion.

`callback`

Fonction à appeler pour traiter le message souscrit.

La `mqtt_connection.subscribe` fonction renvoie un futur et un ID de paquet. Si la demande d'abonnement a été lancée avec succès, l'ID de paquet renvoyé est supérieur à 0. Pour vous assurer que l'abonnement a été reçu et enregistré par le courtier de messages, vous devez attendre le retour du résultat de l'opération asynchrone, comme indiqué dans l'exemple de code.

Fonction de rappel

Le rappel de l'`pubsub.py`exemple traite les messages souscrits au fur et à mesure que l'appareil les reçoit.

```
def on_message_received(topic, payload, **kwargs):
    print("Received message from topic '{}': {}".format(topic, payload))
    global received_count
    received_count += 1
    if received_count == args.count:
        received_all_event.set()
```

`topic`

Le sujet du message

Il s'agit du nom de sujet spécifique du message reçu, même si vous vous êtes abonné à un filtre de sujet.

`payload`

La charge utile des messages

Le format utilisé est spécifique à l'application.

`kwargs`

Arguments supplémentaires possibles tels que décrits dans `mqtt.Connection.subscribe`.

Dans l'`pubsub.py`exemple`on_message_received`, affiche uniquement le sujet et sa charge utile. Il compte également les messages reçus pour terminer le programme une fois la limite atteinte.

Votre application évaluera le sujet et la charge utile pour déterminer les actions à effectuer.

Déconnexion et reconnexion de l'appareil

L'`pubsub.py`exemple inclut des fonctions de rappel qui sont appelées lorsque le périphérique est déconnecté et lorsque la connexion est rétablie. Les actions entreprises par votre appareil face à ces événements sont spécifiques à l'application.

Lorsqu'un appareil se connecte pour la première fois, il doit s'abonner aux rubriques pour pouvoir les recevoir. Si la session d'un appareil est présente lorsqu'il se reconnecte, ses abonnements sont restaurés et tous les messages enregistrés provenant de ces abonnements sont envoyés à l'appareil après sa reconnexion.

Si la session d'un appareil n'existe plus lorsqu'il se reconnecte, il doit se réabonner à ses abonnements. Les sessions persistantes ont une durée de vie limitée et peuvent expirer lorsque l'appareil est déconnecté trop longtemps.

Connectez votre appareil et communiquez avec AWS IoT Core

Cette section présente quelques exercices destinés à vous aider à explorer les différents aspects de la connexion de votre appareil à AWS IoT Core. Pour ces exercices, vous allez utiliser le client de MQTT test de la AWS IoT console pour voir ce que votre appareil publie et pour publier des messages sur votre appareil. Ces exercices utilisent l'`pubsub.py`exemple du AWS IoT Device SDK v2 pour Python et s'appuient sur votre expérience des Didacticiel de démarrage didacticiels.

Dans cette section, vous effectuez les opérations suivantes :

- Abonnez-vous aux filtres thématiques génériques
- Traiter les abonnements aux filtres thématiques
- Publiez des messages depuis votre appareil

Pour ces exercices, vous allez commencer par l'`pubsub.py`exemple de programme.

> **Note**
>
> Ces exercices supposent que vous avez terminé les Didacticiel de démarrage didacticiels et que vous utilisez la fenêtre du terminal de votre appareil à partir de ce didacticiel.

Abonnez-vous aux filtres thématiques génériques

Dans cet exercice, vous allez modifier la ligne de commande utilisée pour `pubsub.py` vous abonner à un filtre de sujet générique et traiter les messages reçus en fonction du sujet du message.

Procédure d'exercice

Pour cet exercice, imaginez que votre appareil contient une commande de température et une commande d'éclairage. Il utilise ces noms de rubriques pour identifier les messages les concernant.

1. Avant de commencer l'exercice, essayez d'exécuter cette commande à partir des Didacticiel de démarrage didacticiels de votre appareil pour vous assurer que tout est prêt pour l'exercice.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
   certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
   ```

   Vous devriez voir le même résultat que celui que vous avez vu dans le didacticiel de démarrage.

2. Pour cet exercice, modifiez ces paramètres de ligne de commande.

Action	Paramètre de ligne de commande	Effet
ajouter	--message ""	Configurer `pubsub.py` pour écouter uniquement
ajouter	--count 2	Terminez le programme après avoir reçu deux messages
modification	--topic `device/+/details`	Définissez le filtre de rubrique auquel vous souhaitez vous abonner

   Ces modifications apportées à la ligne de commande initiale se traduisent par cette ligne de commande. Entrez cette commande dans la fenêtre du terminal de votre appareil.

```
python3 pubsub.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

Le programme doit afficher ce qui suit :

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-24d7cdcc-cc01-458c-8488-2d05849691e1'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
```

Si vous voyez quelque chose comme ça sur votre terminal, celui-ci est prêt et il écoute les messages dont le nom des sujets commence par `device` et se termine par`/detail`. Alors, testons cela.

3. Voici quelques messages que votre appareil est susceptible de recevoir.

Nom de la rubrique	Charge utile du message
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

4. À l'aide du client de MQTT test de la AWS IoT console, envoyez les messages décrits à l'étape précédente à votre appareil.

   a. Ouvrez le client de MQTT test dans la AWS IoT console.

   b. Dans Subscribe to a topic, dans le champ Subscription topic, entrez **`device/+/details`**, puis choisissez Subscribe to topic.

   c. Dans la colonne Abonnements du client de MQTT test, sélectionnez device/+/details.

   d. Pour chacun des sujets du tableau précédent, effectuez les opérations suivantes dans le client de MQTT test :

      1. Dans Publier, entrez la valeur de la colonne Nom du sujet dans le tableau.

2. Dans le champ de charge utile du message situé sous le nom du sujet, entrez la valeur de la colonne Charge utile du message du tableau.
3. Regardez la fenêtre du terminal en cours `pubsub.py` d'exécution et, dans le client de MQTT test, choisissez Publier dans le sujet.

Vous devriez voir que le message a été reçu `pubsub.py` dans la fenêtre du terminal.

Résultat de l'exercice

Ainsi,`pubsub.py`, s'est abonné aux messages à l'aide d'un filtre thématique générique, les a reçus et les a affichés dans la fenêtre du terminal. Notez que vous vous êtes abonné à un filtre de sujet unique et que la fonction de rappel a été appelée pour traiter les messages ayant deux sujets distincts.

Traiter les abonnements aux filtres thématiques

Sur la base de l'exercice précédent, modifiez l'`pubsub.py`exemple d'application pour évaluer les sujets des messages et traiter les messages souscrits en fonction du sujet.

Procédure d'exercice

Pour évaluer le sujet du message

1. Copiez `pubsub.py` dans `pubsub2.py`.
2. Ouvrez `pubsub2.py` dans votre éditeur de texte préféré ouIDE.
3. Dans`pubsub2.py`, trouvez la `on_message_received` fonction.
4. Dans`on_message_received`, insérez le code suivant après la ligne commençant par `print("Received message` et avant la ligne commençant par`global received_count`.

```
    topic_parsed = False
    if "/" in topic:
        parsed_topic = topic.split("/")
        if len(parsed_topic) == 3:
            # this topic has the correct format
            if (parsed_topic[0] == 'device') and (parsed_topic[2] == 'details'):
                # this is a topic we care about, so check the 2nd element
                if (parsed_topic[1] == 'temp'):
                    print("Received temperature request: {}".format(payload))
```

```
                    topic_parsed = True
                if (parsed_topic[1] == 'light'):
                    print("Received light request: {}".format(payload))
                    topic_parsed = True
    if not topic_parsed:
        print("Unrecognized message topic.")
```

5. Enregistrez vos modifications et exécutez le programme modifié à l'aide de cette ligne de commande.

```
python3 pubsub2.py --message "" --count 2 --topic device/+/details --ca_file
 ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/
private.pem.key --endpoint your-iot-endpoint
```

6. Dans la AWS IoT console, ouvrez le client MQTT de test.
7. Dans Subscribe to a topic, dans le champ Subscription topic, entrez **`device/+/details`**, puis choisissez Subscribe to topic.
8. Dans la colonne Abonnements du client de MQTT test, sélectionnez device/+/details.
9. Pour chacune des rubriques de ce tableau, effectuez les opérations suivantes dans le client de MQTT test :

Nom de la rubrique	Charge utile du message
`device/temp/details`	`{ "desiredTemp": 20, "currentTemp": 15 }`
`device/light/details`	`{ "desiredLight": 100, "currentLight": 50 }`

   1. Dans Publier, entrez la valeur de la colonne Nom du sujet dans le tableau.
   2. Dans le champ de charge utile du message situé sous le nom du sujet, entrez la valeur de la colonne Charge utile du message du tableau.
   3. Regardez la fenêtre du terminal en cours `pubsub.py` d'exécution et, dans le client de MQTT test, choisissez Publier dans le sujet.

Vous devriez voir que le message a été reçu `pubsub.py` dans la fenêtre du terminal.

Le résultat devrait être similaire dans la fenêtre de votre terminal.

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID 'test-
af794be0-7542-45a0-b0af-0b0ea7474517'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
Received message from topic 'device/light/details': b'{ "desiredLight": 100,
 "currentLight": 50 }'
Received light request: b'{ "desiredLight": 100, "currentLight": 50 }'
Received message from topic 'device/temp/details': b'{ "desiredTemp": 20,
 "currentTemp": 15 }'
Received temperature request: b'{ "desiredTemp": 20, "currentTemp": 15 }'
2 message(s) received.
Disconnecting...
Disconnected!
```

Résultat de l'exercice

Dans cet exercice, vous avez ajouté du code afin que l'exemple d'application reconnaisse et traite plusieurs messages dans la fonction de rappel. Ainsi, votre appareil pourrait recevoir des messages et agir en conséquence.

Un autre moyen pour votre appareil de recevoir et de traiter plusieurs messages consiste à s'abonner à différents messages séparément et à attribuer à chaque abonnement sa propre fonction de rappel.

Publiez des messages depuis votre appareil

Vous pouvez utiliser l'exemple d'application pubsub.py pour publier des messages depuis votre appareil. Bien qu'il publie les messages tels quels, ils ne peuvent pas être lus sous forme de JSON documents. Cet exercice modifie l'exemple d'application afin de pouvoir publier JSON des documents lisibles dans la charge utile des messages. AWS IoT Core

Procédure d'exercice

Dans cet exercice, le message suivant sera envoyé avec le `device/data` sujet.

```
{
    "timestamp": 1601048303,
    "sensorId": 28,
    "sensorData": [
        {
```

```
        "sensorName": "Wind speed",
        "sensorValue": 34.2211224
        }
    ]
}
```

Pour préparer votre client de MQTT test à surveiller les messages de cet exercice

1. Dans Subscribe to a topic, dans le champ Subscription topic, entrez **`device/data`**, puis choisissez Subscribe to topic.
2. Dans la colonne Abonnements du client de MQTT test, sélectionnez appareil/données.
3. Laissez la fenêtre du client de MQTT test ouverte pour attendre les messages de votre appareil.

Pour envoyer JSON des documents avec l'exemple d'application pubsub.py

1. Sur votre appareil, copiez-le `pubsub.py` vers`pubsub3.py`.
2. Modifiez `pubsub3.py` pour modifier le format des messages qu'il publie.
   a. Ouvrez `pubsub3.py` dans un éditeur de texte.
   b. Localisez cette ligne de code :

      ```
      message = "{} [{}]".format(message_string, publish_count)
      ```

   c. Remplacer par :

      ```
      message = "{}".format(message_string)
      ```

   d. Localisez cette ligne de code :

      ```
      message_json = json.dumps(message)
      ```

   e. Remplacer par :

      ```
      message = "{}".json.dumps(json.loads(message))
      ```

   f. Enregistrez vos modifications.
3. Sur votre appareil, exécutez cette commande pour envoyer le message deux fois.

   ```
   python3 pubsub3.py  --ca_file ~/certs/Amazon-root-CA-1.pem  --cert ~/certs/
   device.pem.crt  --key ~/certs/private.pem.key  --topic device/data  --count 2 --
   message '{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
    speed","sensorValue":34.2211224}]}'  --endpoint your-iot-endpoint
   ```

4. Dans le client de MQTT test, vérifiez qu'il a interprété et formaté le JSON document dans la charge utile du message, par exemple :

Par défaut, s'abonne `pubsub3.py` également aux messages qu'il envoie. Vous devriez voir qu'elle a reçu les messages dans la sortie de l'application. La fenêtre du terminal doit se présenter comme suit.

```
Connecting to a3qEXAMPLEsffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-5cff18ae-1e92-4c38-a9d4-7b9771afc52f'...
Connected!
Subscribing to topic 'device/data'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
2 message(s) received.
Disconnecting...
Disconnected!
```

Résultat de l'exercice

Ainsi, votre appareil peut générer des messages à envoyer AWS IoT Core pour tester la connectivité de base et fournir des messages AWS IoT Core à traiter. Par exemple, vous pouvez utiliser cette application pour envoyer des données de test depuis votre appareil afin de tester les actions des AWS IoT règles.

Passez en revue les résultats

Les exemples présentés dans ce didacticiel vous ont permis d'acquérir une expérience pratique des bases de la communication entre les AWS IoT Core appareils, élément fondamental de votre AWS IoT solution. Lorsque vos appareils sont en mesure de communiquer avec eux AWS IoT Core, ils peuvent transmettre des messages aux AWS services et autres appareils sur lesquels ils peuvent agir. De même, AWS les services et autres appareils peuvent traiter des informations qui se traduisent par le renvoi de messages vers vos appareils.

Lorsque vous serez prêt à AWS IoT Core poursuivre votre exploration, essayez ces didacticiels :

- the section called “Envoi d'une SNS notification Amazon”
- the section called “Stockage des données de l'appareil dans une table DynamoDB”
- the section called “Formatage d'une notification à l'aide d'une AWS Lambda fonction”

Tutoriel : Utilisation du Kit SDK des appareils AWS IoT pour Embedded C

Cette section décrit comment exécuter le Kit SDK des appareils AWS IoT pour Embedded C.

Procédures décrites dans cette section

- Étape 1 : installez le Kit SDK des appareils AWS IoT pour Embedded C
- Étape 2 : Configurer l'exemple d'application
- Étape 3 : Créer et exécuter l'exemple d'application

Étape 1 : installez le Kit SDK des appareils AWS IoT pour Embedded C

Kit SDK des appareils AWS IoT pour Embedded C Il est généralement destiné aux appareils aux ressources limitées qui nécessitent un environnement d'exécution optimisé en langage C. Vous pouvez l'utiliser SDK sur n'importe quel système d'exploitation et l'héberger sur n'importe quel type de processeur (par exemple, MCUs etMPUs). Si vous disposez de davantage de mémoire et de

ressources de traitement, nous vous recommandons d'utiliser un AWS IoT appareil ou un appareil mobile de niveau supérieur SDKs (par exemple, C++ JavaScript, Java et Python).

En général, Kit SDK des appareils AWS IoT pour Embedded C il est destiné aux systèmes utilisant MCUs ou bas de gamme exécutant MPUs des systèmes d'exploitation intégrés. Pour l'exemple de programmation présenté dans cette section, nous supposons que votre appareil utilise Linux.

Example

1. Kit SDK des appareils AWS IoT pour Embedded C Téléchargez-le sur votre appareil depuis GitHub.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-embedded-c.git --recurse-
   submodules
   ```

   Cette opération crée un répertoire nommé `aws-iot-device-sdk-embedded-c` dans le répertoire actuel.

2. Accédez à ce répertoire et consultez la dernière version. Consultez github.com/aws/ aws-iot-device-sdk -Embedded-c/tags pour la dernière balise de version.

   ```
   cd aws-iot-device-sdk-embedded-c
   git checkout latest-release-tag
   ```

3. Installez Open SSL version 1.1.0 ou ultérieure. Les bibliothèques de SSL développement ouvertes sont généralement appelées « libssl-dev » ou « openssl-devel » lorsqu'elles sont installées via un gestionnaire de paquets.

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

Étape 2 : Configurer l'exemple d'application

Kit SDK des appareils AWS IoT pour Embedded C Il inclut des exemples d'applications que vous pouvez essayer. Pour des raisons de simplicité, ce didacticiel utilise l'`mqtt_demo_mutual_auth`application, qui montre comment se connecter au courtier de AWS IoT Core messages, s'abonner et publier dans MQTT des rubriques.

1. Copiez le certificat et la clé privée que vous avez créés dans Commencer à utiliser les AWS IoT Core didacticiels dans le répertoire `build/bin/certificates`.

> **Note**
>
> Les certificats d'autorité de certification racine et d'appareil sont susceptibles d'expirer ou d'être révoqués. Si ces certificats expirent ou sont révoqués, vous devez copier un nouveau certificat d'autorité de certification ou une nouvelle clé privée et un nouveau certificat d'appareil sur votre appareil.

2. Vous devez configurer l'exemple avec votre point de AWS IoT Core terminaison personnel, votre clé privée, votre certificat et votre certificat CA racine. Accédez au répertoire `aws-iot-device-sdk-embedded-c/demos/mqtt/mqtt_demo_mutual_auth`.

   Si vous l'avez AWS CLI installé, vous pouvez utiliser cette commande pour trouver le point de terminaison de votre compteURL.

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

   Si ce n'est pas le AWS CLI cas, ouvrez votre AWS IoT console. Dans le panneau de navigation, choisissez Manage (Gérer), puis Things (Objets). Choisissez l'objet IoT pour votre appareil, puis choisissez Interagir. Votre point de terminaison s'affiche dans la HTTPSsection de la page des détails de l'objet.

3. Ouvrez le fichier `demo_config.h` et mettez à jour les valeurs des éléments suivants :

   AWS_IOT_ENDPOINT

   Votre point de terminaison personnel.

   CLIENT_CERT_PATH

   Le chemin de votre fichier de certificat, par exemple `certificates/device.pem.crt`".

   CLIENT_PRIVATE_KEY_PATH

   Le nom de votre fichier de clé privée, par exemple`certificates/private.pem.key`.

   Par exemple :

   ```
   // Get from demo_config.h
   // ================================================
   ```

```
#define AWS_IOT_ENDPOINT                "my-endpoint-ats.iot.us-
east-1.amazonaws.com"
#define AWS_MQTT_PORT                   8883
#define CLIENT_IDENTIFIER               "testclient"
#define ROOT_CA_CERT_PATH               "certificates/AmazonRootCA1.crt"
#define CLIENT_CERT_PATH                "certificates/my-device-cert.pem.crt"
#define CLIENT_PRIVATE_KEY_PATH         "certificates/my-device-private-key.pem.key"
// ================================================
```

4. Vérifiez si vous l'avez CMake installé sur votre appareil à l'aide de cette commande.

   ```
   cmake --version
   ```

   Si vous voyez les informations de version du compilateur, vous pouvez passer à la section suivante.

   Si vous obtenez une erreur ou ne voyez aucune information, vous devrez alors installer le package cmake à l'aide de cette commande.

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

   Réexécutez la cmake --version commande et confirmez qu'CMakeelle a été installée et que vous êtes prêt à continuer.

5. Vérifiez si les outils de développement sont installés sur votre appareil à l'aide de cette commande.

   ```
   gcc --version
   ```

   Si vous voyez les informations de version du compilateur, vous pouvez passer à la section suivante.

   Si vous obtenez une erreur ou que vous ne voyez aucune information de compilateur, vous devrez installer le package `build-essential` à l'aide de cette commande.

   ```
   sudo apt-get install build-essential
   ```

   Exécutez à nouveau la commande gcc --version et vérifiez que les outils de génération ont été installés et que vous êtes prêt à continuer.

Étape 3 : Créer et exécuter l'exemple d'application

Cette procédure explique comment générer l'`mqtt_demo_mutual_auth`application sur votre appareil et comment la connecter à la AWS IoT console à l'aide du. Kit SDK des appareils AWS IoT pour Embedded C

Pour exécuter les Kit SDK des appareils AWS IoT pour Embedded C exemples d'applications

1. Accédez à un répertoire de construction `aws-iot-device-sdk-embedded-c` et créez-le.

   ```
   mkdir build && cd build
   ```

2. Entrez la CMake commande suivante pour générer les Makefiles nécessaires à la construction.

   ```
   cmake ..
   ```

3. Entrez la commande suivante pour créer le fichier d'application exécutable.

   ```
   make
   ```

4. Exécutez l'application `mqtt_demo_mutual_auth` avec cette commande.

   ```
   cd bin
   ./mqtt_demo_mutual_auth
   ```

   Vous devez voir des résultats similaires à ce qui suit :

```
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:584] Establishing a TLS session to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com:8883.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1264] Creating an MQTT connection to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com.
[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt_serializer.c:970] CONNACK session present bit not set.
[INFO] [MQTT] [core_mqtt_serializer.c:912] Connection accepted.
[INFO] [MQTT] [core_mqtt.c:1526] Received MQTT CONNACK successfully from broker.
[INFO] [MQTT] [core_mqtt.c:1792] MQTT connection established with the broker.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1033] MQTT connection successfully established with broker.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1296] A clean MQTT connection is established. Cleaning up all the stored outgoing publishes.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1314] Subscribing to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1097] SUBSCRIBE sent for topic testclient/example/topic to broker.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=3.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:921] Subscribed to the topic testclient/example/topic. with maximum QoS 1.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1358] Sending Publish to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1195] PUBLISH sent for topic testclient/example/topic to broker with packet ID 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt.c:1126] Ack packet deserialized with result: MQTTSuccess.
[INFO] [MQTT] [core_mqtt.c:1139] State record updated. New state=MQTTPublishDone.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:946] PUBACK received for packet id 2.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:672] Cleaned up outgoing publish packet with packet id 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=40.
[INFO] [MQTT] [core_mqtt.c:1015] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
```

Votre appareil est maintenant connecté à AWS IoT l'aide du Kit SDK des appareils AWS IoT pour Embedded C.

Vous pouvez également utiliser la AWS IoT console pour afficher les MQTT messages publiés par l'exemple d'application. Pour plus d'informations sur l'utilisation du MQTT client dans la AWS IoT console, consultezthe section called “Afficher les messages MQTT avec le client AWS IoT MQTT”.

Création de AWS IoT règles pour acheminer les données des appareils vers d'autres services

Ces didacticiels vous montrent comment créer et tester des AWS IoT règles à l'aide de certaines des actions de règles les plus courantes.

AWS IoT les règles envoient des données de vos appareils à d'autres AWS services. Ils écoutent des MQTT messages spécifiques, mettent en forme les données contenues dans les charges utiles des messages et envoient le résultat à d'autres AWS services.

Nous vous recommandons de les essayer dans l'ordre dans lequel ils apparaissent ici, même si votre objectif est de créer une règle utilisant une fonction Lambda ou une fonction plus complexe.

Les didacticiels sont présentés dans l'ordre du plus élémentaire au plus complexe. Ils présentent de nouveaux concepts de manière progressive afin de vous aider à apprendre les concepts que vous pouvez utiliser pour créer les actions de règles qui ne font pas l'objet d'un didacticiel spécifique.

> **Note**
>
> AWS IoT les règles vous aident à envoyer les données de vos appareils IoT vers d'autres AWS services. Pour le faire avec succès, vous devez toutefois avoir une connaissance pratique des autres services auxquels vous souhaitez envoyer des données. Bien que ces didacticiels fournissent les informations nécessaires pour effectuer les tâches, vous trouverez peut-être utile d'en savoir plus sur les services auxquels vous souhaitez envoyer des données avant de les utiliser dans votre solution. Une explication détaillée des autres AWS services n'entre pas dans le cadre de ces didacticiels.

Présentation des scénarios de didacticiel

Le scénario de ces didacticiels est celui d'un capteur météo qui publie périodiquement ses données. Il existe de nombreux capteurs de ce type dans ce système imaginaire. Les didacticiels de cette section se concentrent toutefois sur un seul appareil tout en montrant comment vous pouvez intégrer plusieurs capteurs.

Les didacticiels de cette section vous montrent comment utiliser des AWS IoT règles pour effectuer les tâches suivantes avec ce système imaginaire de capteurs météorologiques.

- Tutoriel : Republication d'un message MQTT

  Ce didacticiel montre comment republier un MQTT message reçu des capteurs météorologiques sous la forme d'un message contenant uniquement l'identifiant du capteur et la valeur de température. Il utilise uniquement AWS IoT Core des services et montre une SQL requête simple et explique comment utiliser le MQTT client pour tester votre règle.

- Tutoriel : Envoi d'une SNS notification Amazon

  Ce didacticiel explique comment envoyer un SNS message lorsqu'une valeur d'un capteur météo dépasse une valeur spécifique. Il s'appuie sur les concepts présentés dans le didacticiel précédent et explique comment travailler avec un autre AWS service, l'Amazon Simple Notification Service (AmazonSNS).

Si vous utilisez Amazon pour la première SNS fois, consultez ses exercices de démarrage avant de commencer ce didacticiel.

- Tutoriel : Stockage des données de l'appareil dans une table DynamoDB

  Ce didacticiel montre comment stocker les données des capteurs météorologiques dans une table de base de données. Il utilise l'énoncé de requête de règle et les modèles de substitution pour formater les données des messages pour le service de destination, Amazon DynamoDB.

  Si vous utilisez DynamoDB pour la première fois, consultez ses exercices de mise en route avant de commencer ce didacticiel.

- Tutoriel : Formatage d'une notification à l'aide d'une AWS Lambda fonction

  Ce didacticiel explique comment appeler une fonction Lambda pour reformater les données de l'appareil, puis les envoyer sous forme de message texte. Il ajoute un script Python et des AWS SDK fonctions dans une AWS Lambdafonction pour formater avec le message les données de charge utile provenant des capteurs météorologiques et envoyer un message texte.

  Si vous utilisez Lambda pour la première fois, consultez ses exercices de mise en route avant de commencer ce didacticiel.

AWS IoT vue d'ensemble des règles

Tous ces didacticiels créent des AWS IoT règles.

Pour qu'une AWS IoT règle envoie les données d'un appareil à un autre AWS service, elle utilise :

- Une déclarationde requête de règle composée des éléments suivants :
  - Une SQL SELECT clause qui sélectionne et met en forme les données de la charge utile du message
  - Un filtre de rubrique (l'FROMobjet de l'instruction de requête de règle) qui identifie les messages à utiliser
  - Une déclaration conditionnelle facultative (une SQL WHERE clause) qui spécifie les conditions spécifiques sur lesquelles agir
- Au moins une action de règle

Les appareils publient des messages dans MQTT des rubriques. Le filtre de rubrique de la SQL SELECT déclaration identifie les MQTT sujets auxquels appliquer la règle. Les champs spécifiés dans l'SQLSELECTinstruction mettent en forme les données de la charge utile des MQTT messages entrants afin qu'elles soient utilisées par les actions de la règle. Pour obtenir la liste complète des actions de règle, consultez Actions de règle AWS IoT.

Didacticiels dans cette section

- Tutoriel : Republication d'un message MQTT
- Tutoriel : Envoi d'une SNS notification Amazon
- Tutoriel : Stockage des données de l'appareil dans une table DynamoDB
- Tutoriel : Formatage d'une notification à l'aide d'une AWS Lambda fonction

Tutoriel : Republication d'un message MQTT

Ce didacticiel explique comment créer une AWS IoT règle qui publie un MQTT message lorsqu'un MQTT message spécifique est reçu. La charge utile des messages entrants peut être modifiée par la règle avant sa publication. Cela permet de créer des messages adaptés à des applications spécifiques sans qu'il soit nécessaire de modifier votre appareil ou son microprogramme. Vous pouvez également utiliser l'aspect filtrage d'une règle pour publier des messages uniquement lorsqu'une condition spécifique est remplie.

Les messages republiés par une règle agissent comme des messages envoyés par n'importe quel autre AWS IoT appareil ou client. Les appareils peuvent s'abonner aux messages republiés de la même manière qu'ils peuvent s'abonner à n'importe quel autre sujet de MQTT message.

Ce que vous allez apprendre dans ce didacticiel:

- Comment utiliser des SQL requêtes et des fonctions simples dans une instruction de requête de règle
- Comment utiliser le MQTT client pour tester une AWS IoT règle

Ce didacticiel vous prendra environ 30 minutes.

Dans ce tutoriel, vous allez :

- Réviser MQTT les sujets et AWS IoT les règles
- Étape 1 : créer une AWS IoT règle pour republier un message MQTT

- Étape 2 : Test de votre règle
- Étape 3 : examen des résultats et des étapes suivantes

Avant de commencer ce didacticiel, assurez-vous de disposer des éléments suivants :

- Configurez Compte AWS

  Vous aurez besoin de votre AWS IoT console Compte AWS et de votre console pour terminer ce didacticiel.
- Révisé Afficher les messages MQTT avec le client AWS IoT MQTT

  Assurez-vous que vous pouvez utiliser le MQTT client pour vous abonner à un sujet et le publier. Vous allez utiliser le MQTT client pour tester votre nouvelle règle dans le cadre de cette procédure.

Réviser MQTT les sujets et AWS IoT les règles

Avant de parler de AWS IoT règles, il est utile de comprendre le MQTT protocole. Dans les solutions IoT, le MQTT protocole offre certains avantages par rapport aux autres protocoles de communication réseauHTTP, ce qui en fait notamment un choix populaire pour les appareils IoT. Cette section passe en revue les principaux aspects de ce didacticiel MQTT tels qu'ils s'appliquent à ce didacticiel. Pour plus d'informations sur le MQTT mode de comparaison avecHTTP, voirChoix d'un protocole d'application pour la communication de votre appareil.

MQTTprotocole

Le MQTT protocole utilise un modèle de communication de publication/abonnement avec son hôte. Pour envoyer des données, les appareils publient des messages identifiés par des sujets sur le courtier de AWS IoT messages. Pour recevoir des messages du courtier de messages, les appareils s'abonnent aux objets qu'ils recevront en envoyant des filtres de objets dans les demandes d'abonnement adressées au courtier de messages. Le moteur de AWS IoT règles reçoit MQTT les messages du courtier de messages.

AWS IoT règles

AWS IoT les règles consistent en une instruction de requête de règle et une ou plusieurs actions de règles. Lorsque le moteur de AWS IoT règles reçoit un MQTT message, ces éléments agissent sur le message comme suit.

- Déclaration de requête de règle

L'instruction de requête de la règle décrit les MQTT sujets à utiliser, interprète les données issues de la charge utile du message et met en forme les données conformément à une SQL instruction similaire aux instructions utilisées par les bases de données courantesSQL. Le résultat de l'déclararionde requête correspond aux données envoyées aux actions de la règle.

- Action de la règle

  Chaque action de règle d'une règle agit sur les données résultant de l'instruction de requête de la règle. AWS IoT prend en charge de nombreuses actions de règles. Dans ce didacticiel, vous allez toutefois vous concentrer sur l'action de la Republish règle, qui publie le résultat de l'instruction de requête sous forme de MQTT message avec un sujet spécifique.

Étape 1 : créer une AWS IoT règle pour republier un message MQTT

La AWS IoT règle que vous allez créer dans ce didacticiel s'applique aux `device/`*`device_id`*`/data` MQTT rubriques où *`device_id`* est l'identifiant de l'appareil qui a envoyé le message. Ces rubriques sont décrites par un filtre de rubrique en tan que `device/+/data`, où le + est un caractère générique correspondant à n'importe quelle chaîne située entre les deux barres obliques.

Lorsque la règle reçoit un message d'un sujet correspondant, elle republie les `temperature` valeurs `device_id` et sous forme de nouveau MQTT message avec le `device/data/temp` sujet.

Par exemple, la charge utile d'un MQTT message contenant le `device/22/data` sujet se présente comme suit :

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

La règle prend la `temperature` valeur de la charge utile du message et celle `device_id` du sujet, et les republie sous forme de MQTT message avec le `device/data/temp` sujet et une charge utile du message qui ressemble à ceci :

```
{
```

```
  "device_id": "22",
  "temperature": 28
}
```

Avec cette règle, les appareils qui n'ont besoin que de l'identifiant de l'appareil et des données de température s'abonnent à la `device/data/temp` rubrique pour ne recevoir que ces informations.

Pour créer une règle qui republie un message MQTT

1. Ouvrez le hub de règles de la AWS IoT console.
2. Dans Règles, choisissez Créer et commencez à créer votre nouvelle règle.
3. Dans la partie supérieure de Créer une règle :

   a. Dans Nom, entrez le nom de la règle. Pour ce didacticiel, nommez le **`republish_temp`**.

      N'oubliez pas qu'un nom de règle doit être unique au sein de votre compte et de votre région, et qu'il ne doit pas comporter d'espaces. Nous avons utilisé un trait de soulignement dans ce nom pour séparer les deux mots du nom de la règle.

   b. Dans Description, décrivez la règle.

      Une description significative vous permet de vous souvenir du rôle de cette règle et de la raison pour laquelle vous l'avez créée. La description peut être aussi longue que nécessaire, donc soyez aussi détaillée que possible.
4. Dans l'déclararionde requête règle de Create a rule :

   a. Dans Utiliser SQL la version, sélectionnez**`2016-03-23`**.

   b. Dans la zone d'édition de la instruction de requête de règle, entrez l'instruction :

      ```
      SELECT topic(2) as device_id, temperature FROM 'device/+/data'
      ```

      Cette instruction :

      - Écoute les MQTT messages dont le sujet correspond au filtre de `device/+/data` sujet.
      - Sélectionne le deuxième élément dans la chaîne de l'objet et l'affecte au `device_id` champ.
      - Sélectionne le `temperature` champ de valeur dans la charge utile du message et l'affecte au `temperature` champ.
5. Dans Définir une ou plusieurs actions :

a. Pour ouvrir la liste des actions de règle pour cette règle, choisissez Ajouter une action.

b. Dans Sélectionner une action, choisissez Republier un message dans un AWS IoT sujet.

c. Au bas de la liste d'actions, choisissez Configurer l'action pour ouvrir la page de configuration de l'action sélectionnée.

6. Sous Configurer les actions :

a. Dans Sujet, entrez **`device/data/temp`**. C'est le MQTT sujet du message que cette règle publiera.

b. Dans Qualité de service, choisissez 0 - Le message est délivré zéro fois ou plus.

c. Dans Choisir ou créer un rôle pour accorder AWS IoT l'accès pour effectuer cette action :

i. Choisissez Create Role (Créer le rôle). La boîte de dialogue Créer un rôle s'ouvre.

ii. Saisissez un nom qui décrit le nouveau rôle. Dans le cadre de ce tutoriel, utilisez **`republish_role`**.

Lorsque vous créez un nouveau rôle, les politiques appropriées pour exécuter l'action de la règle sont créées et associées au nouveau rôle. Si vous modifiez le objet de cette action de règle ou si vous utilisez ce rôle dans une autre action de règle, vous devez mettre à jour la politique de ce rôle afin d'autoriser le nouveau objet ou la nouvelle action. Pour mettre à jour un rôle existant, choisissez Mettre à jour le rôle dans cette section.

iii. Choisissez Create Role pour créer le rôle et fermer la boîte de dialogue.

d. Choisissez Ajouter une action pour ajouter l'action à la règle et revenez à la page Créer une règle.

7. L'action Republier un message dans un AWS IoT sujet est désormais répertoriée dans Définir une ou plusieurs actions.

Dans la vignette de la nouvelle action, sous Republier un message dans un AWS IoT objet, vous pouvez voir le objet dans lequel votre action de republication sera publiée.

Il s'agit de la seule action de règle que vous ajouterez à cette règle.

8. Dans Créer une règle, faites défiler l'écran vers le bas et choisissez Créer une règle pour créer la règle et terminer cette étape.

Étape 2 : Test de votre règle

Pour tester votre nouvelle règle, vous allez utiliser le MQTT client pour publier les MQTT messages utilisés par cette règle et vous y abonner.

Ouvrez le MQTTclient dans la AWS IoT console dans une nouvelle fenêtre. Cela vous permettra de modifier la règle sans perdre la configuration de votre MQTT client. Le MQTT client ne conserve aucun abonnement ou journal des messages si vous le quittez pour accéder à une autre page de la console.

Pour utiliser le MQTT client pour tester votre règle

1. Dans le MQTTclient de la AWS IoT console, abonnez-vous aux sujets d'entrée, dans ce cas,`device/+/data`.

   a. Dans le MQTT client, sous Abonnements, choisissez S'abonner à un sujet.

   b. Dans Sujet d'abonnement, entrez le objet du filtre de objet d'entrée,**`device/+/data`**.

   c. Conservez les valeurs par défaut des autres paramètres.

   d. Choisissez Subscribe to topic (S'abonner à la rubrique).

   Dans la colonne Abonnements, la section Publier dans un objet **`device/+/data`** apparaît.

2. Abonnez-vous à l'objet que votre règle publiera :`device/data/temp`.

   a. Sous Abonnements, choisissez S'abonner à un objet nouveau, puis dans Sujet d'abonnement, entrez l'objet du message republié,**`device/data/temp`**.

   b. Conservez les paramètres par défaut du reste des champs

   c. Choisissez Subscribe to topic (S'abonner à la rubrique).

   Dans la colonne Abonnements, la section appareil/+/data, **`device/data/temp`** apparaît.

3. Publiez un message sur le objet d'entrée avec un identifiant d'appareil spécifique, **`device/22/data`**. Vous ne pouvez pas publier dans MQTT des rubriques contenant des caractères génériques.

   a. Dans le MQTT client, sous Abonnements, choisissez Publier dans le sujet.

   b. Dans le champ Publier, entrez le nom de l'objet d'entrée, **`device/22/data`**.

   c. Copiez les exemples de données présentés ici et, dans la zone d'édition située sous le nom de l'objet, collez les exemples de données.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Pour envoyer votre MQTT message, choisissez Publier dans le sujet.

4. Passez en revue les messages qui ont été envoyés.

a. Dans le MQTT client, sous Abonnements, il y a un point vert à côté des deux sujets auxquels vous vous êtes abonné précédemment.

Les points verts indiquent qu'un ou plusieurs nouveaux messages ont été reçus depuis la dernière fois que vous les avez consultés.

b. Sous Abonnements, choisissez device/+/data pour vérifier que la charge utile du message correspond à ce que vous venez de publier et ressemble à ceci :

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Sous Abonnements, choisissez device/data/temp pour vérifier que la charge utile de votre message republié ressemble à ceci :

```
{
  "device_id": "22",
  "temperature": 28
}
```

Notez que la `device_id` valeur est une chaîne entre guillemets et que la `temperature` valeur est numérique. Cela est dû au fait que la `topic()` fonction a extrait la chaîne du nom de l'objet du message d'entrée tandis que la `temperature` valeur utilise la valeur numérique de la charge utile du message d'entrée.

Si vous souhaitez transformer la `device_id` valeur en valeur numérique, remplacez-la `topic(2)` dans l'déclararionde requête de règle par :

```
cast(topic(2) AS DECIMAL)
```

Notez que la conversion de la `topic(2)` valeur en valeur numérique ne fonctionnera que si cette partie de la rubrique contient uniquement des caractères numériques.

5. Si vous constatez que le message correct a été publié dans la rubrique device/data/temp, votre règle a fonctionné. Découvrez les informations supplémentaires que vous pouvez obtenir sur l'action Republier la règle dans la section suivante.

   Si vous ne voyez pas que le message correct a été publié dans les rubriques device/+/data ou device/data/temp, consultez les conseils de dépannage.

Résolution des problèmes liés à la règle de republication des messages

Voici quelques points à vérifier au cas où vous n'obtiendriez pas les résultats escomptés.

- Vous avez reçu une bannière d'erreur

  Si une erreur est apparue lorsque vous avez publié le message d'entrée, corrigez-la d'abord. Les étapes suivantes peuvent vous aider à corriger cette erreur.

- Vous ne voyez pas le message d'entrée dans le MQTT client

  Chaque fois que vous publiez votre message d'entrée dans le `device/22/data` sujet, ce message doit apparaître dans le MQTT client si vous vous êtes abonné au filtre de `device/+/data` sujet comme décrit dans la procédure.

  À savoir

  - Vérifiez le filtre de objets auquel vous vous êtes abonné

Si vous vous êtes abonné au objet du message d'entrée comme décrit dans la procédure, vous devriez voir une copie du message d'entrée chaque fois que vous le publiez.

Si le message ne s'affiche pas, vérifiez le nom de l'objet auquel vous vous êtes abonné et comparez-le au objet dans lequel vous avez publié. Les noms des objets distinguent les majuscules et minuscules et le objet auquel vous vous êtes abonné doit être identique au objet dans lequel vous avez publié la charge utile du message.

- Vérifiez la fonction de publication des messages

  Dans le MQTT client, sous Abonnements, choisissez device/+/data, vérifiez le sujet du message de publication, puis choisissez Publier dans le sujet. La charge utile du message figurant dans la zone d'édition située sous le objet devrait apparaître dans la liste des messages.

- Vous ne voyez pas votre message republié dans le client MQTT

  Pour que votre règle fonctionne, elle doit disposer de la politique appropriée qui l'autorise à recevoir et à republier un message et elle doit recevoir le message.

  À savoir

  - Vérifiez le Région AWS nom de votre MQTT client et la règle que vous avez créée

    La console dans laquelle vous exécutez le MQTT client doit se trouver dans la même AWS région que la règle que vous avez créée.

  - Vérifiez le objet du message d'entrée dans la déclaration de requête de règle

    Pour que la règle fonctionne, elle doit recevoir un message dont le nom de rubrique correspond au filtre de rubrique figurant dans la FROM clause de l'instruction de requête de règle.

    Vérifiez l'orthographe du filtre de rubrique dans l'instruction de requête de règle avec celle du sujet dans le MQTT client. Les noms de objets distinguent les majuscules et minuscules et le objet du message doit correspondre au filtre de objet indiqué dans l'déclararionde requête de règle.

  - Vérifiez le contenu de la charge utile des messages d'entrée

    Pour que la règle fonctionne, elle doit trouver le champ de données dans la charge utile du message déclarée dans l'SELECTinstruction.

    Vérifiez l'orthographe du `temperature` champ dans l'instruction de requête de règle avec celle de la charge utile du message dans le MQTT client. Les noms de champs distinguent les

majuscules et minuscules et le `temperature` champ de l'déclararionde requête de règle doit être identique au `temperature` champ de la charge du message.

Assurez-vous que le JSON document contenu dans la charge utile du message est correctement formaté. S'il JSON contient des erreurs, telles qu'une virgule manquante, la règle ne pourra pas le lire.

- Vérifiez le objet du message republié dans l'action de la règle

  Le sujet auquel l'action Republier la règle publie le nouveau message doit correspondre au sujet auquel vous vous êtes abonné dans le MQTT client.

  Ouvrez la règle que vous avez créée dans la console et vérifiez le objet dans lequel l'action de règle republiera le message.

- Vérifiez le rôle utilisé par la règle

  L'action de règle doit être autorisée à recevoir le objet d'origine et à publier le nouveau objet.

  Les politiques qui autorisent la règle à recevoir les données des messages et à les republier sont spécifiques aux objets utilisés. Si vous modifiez le objet utilisé pour republier les données du message, vous devez mettre à jour le rôle de l'action de règle afin de mettre à jour sa politique afin qu'elle corresponde au objet actuel.

  Si vous pensez que c'est le problème, modifiez l'action Republier la règle et créez un nouveau rôle. Les nouveaux rôles créés par l'action de règle reçoivent les autorisations nécessaires pour effectuer ces actions.

Étape 3 : examen des résultats et des étapes suivantes

Dans ce tutoriel

- Vous avez utilisé une SQL requête simple et quelques fonctions dans une instruction de requête de règle pour générer un nouveau MQTT message.
- Vous avez créé une règle qui republie ce nouveau message.
- Vous avez utilisé le MQTT client pour tester votre AWS IoT règle.

Étapes suivantes

Après avoir republié quelques messages avec cette règle, essayez de l'utiliser pour voir comment la modification de certains aspects du didacticiel affecte le message republié. Voici quelques idées pour vous aider à démarrer.

- Changez le *`device_id`* dans le sujet du message d'entrée et observez l'effet sur la charge utile du message republié.
- Modifiez les champs sélectionnés dans l'déclararionde requête de règle et observez l'effet sur la charge utile des messages republiés.
- Essayez le prochain didacticiel de cette série et découvrez comment Tutoriel : Envoi d'une SNS notification Amazon.

L'action Republier la règle utilisée dans ce didacticiel peut également vous aider à déboguer les instructions de requête relatives aux règles. Par exemple, vous pouvez ajouter cette action à une règle pour voir comment son déclararionde requête de règle met en forme les données utilisées par ses actions de règle.

Tutoriel : Envoi d'une SNS notification Amazon

Ce didacticiel explique comment créer une AWS IoT règle qui envoie des données de MQTT message à un SNS sujet Amazon afin qu'elles puissent être envoyées sous forme de SMS SMS.

Dans ce didacticiel, vous allez créer une règle qui envoie des données de message depuis un capteur météo à tous les abonnés d'un SNS sujet Amazon, chaque fois que la température dépasse la valeur définie dans la règle. La règle détecte lorsque la température signalée dépasse la valeur définie par la règle, puis crée une nouvelle charge utile de message qui inclut uniquement l'identifiant de l'appareil, la température signalée et la limite de température dépassée. La règle envoie la charge utile du nouveau message sous forme de JSON document à une SNS rubrique, qui en informe tous les SNS abonnés.

Ce que vous allez apprendre dans ce didacticiel:

- Comment créer et tester une SNS notification Amazon
- Comment appeler une SNS notification Amazon à partir d'une AWS IoT règle
- Comment utiliser des SQL requêtes et des fonctions simples dans une instruction de requête de règle
- Comment utiliser le MQTT client pour tester une AWS IoT règle

Ce didacticiel vous prendra environ 30 minutes.

Dans ce tutoriel, vous allez :

- Étape 1 : créer un SNS sujet Amazon qui envoie un message SMS texte
- Étape 2 : créer une AWS IoT règle pour envoyer le message texte
- Étape 3 : tester la AWS IoT règle et la SNS notification Amazon
- Étape 4 : examen des résultats et des étapes suivantes

Avant de commencer ce didacticiel, assurez-vous de disposer des éléments suivants :

- Configurez Compte AWS

  Vous aurez besoin de votre AWS IoT console Compte AWS et de votre console pour terminer ce didacticiel.
- Révisé Afficher les messages MQTT avec le client AWS IoT MQTT

  Assurez-vous que vous pouvez utiliser le MQTT client pour vous abonner à un sujet et le publier. Vous allez utiliser le MQTT client pour tester votre nouvelle règle dans le cadre de cette procédure.
- Révisé le Amazon Simple Notification Service

  Si vous n'avez jamais utilisé Amazon SNS auparavant, consultez Configuration de l'accès pour Amazon SNS. Si vous avez déjà suivi d'autres AWS IoT didacticiels, vous Compte AWS devriez déjà être correctement configuré.

Étape 1 : créer un SNS sujet Amazon qui envoie un message SMS texte

Cette procédure explique comment créer le SNS sujet Amazon auquel votre capteur météo peut envoyer des données de message. Le SNS sujet Amazon informera ensuite tous ses abonnés par SMS SMS de la limite de température dépassée.

Pour créer un SNS sujet Amazon qui envoie un message SMS texte

1. Créez un SNS sujet Amazon.

   a. Connectez-vous à la SNSconsole Amazon.

   b. Dans le panneau de navigation de gauche, choisissez Rubriques.

   c. Sur la page Rubriques, choisissez Créer une rubrique.

d. Dans Détails, choisissez le type Standard. Par défaut, la console crée un FIFO sujet.

e. Dans Nom, entrez le nom du SNS sujet. Dans le cadre de ce didacticiel, entrez **`high_temp_notice`**.

f. Faites défiler la page jusqu'en bas et choisissez Créer une rubrique.

La console ouvre la page Détails de la nouvelle rubrique.

2. Créez un SNS abonnement Amazon.

> **Note**
>
> Le numéro de téléphone que vous utilisez dans cet abonnement peut entraîner des frais de messagerie texte en raison des messages que vous allez envoyer dans ce didacticiel.

a. Sur la page des détails de la rubrique high_temp_notice, sélectionnez Créer un abonnement.

b. Dans Créer un abonnement, dans la section Détails, dans la liste des protocoles, sélectionnez SMS.

c. Dans Endpoint, entrez le numéro d'un téléphone pouvant recevoir des SMS. Assurez-vous de le saisir de telle sorte qu'il commence par un +, qu'il inclue le code du pays et de la région, et qu'il n'inclue aucun autre caractère de ponctuation.

d. Choisissez Create subscription (Créer un abonnement).

3. Testez la SNS notification Amazon.

a. Dans la SNSconsole Amazon, dans le volet de navigation de gauche, choisissez Topics.

b. Pour ouvrir la page de détails de l'objet, dans Rubriques, dans la liste des objets, choisissez high_temp_notice.

c. Pour ouvrir la page Publier le message dans le objet, sur la page de détails de high_temp_notice, choisissez Publier le message.

d. Dans Publier le message dans le objet, dans la section Corps du message, dans le corps du message à envoyer au point de terminaison, entrez un message court.

e. Faites défiler la page vers le bas et choisissez Publier des message.

f. Sur le téléphone avec le numéro que vous avez utilisé précédemment lors de la création de l'abonnement, confirmez que le message a bien été reçu.

Si vous n'avez pas reçu le message de test, vérifiez le numéro de téléphone et les paramètres de votre téléphone.

Assurez-vous de pouvoir publier des messages de test depuis la SNSconsole Amazon avant de poursuivre le didacticiel.

Étape 2 : créer une AWS IoT règle pour envoyer le message texte

La AWS IoT règle que vous allez créer dans ce didacticiel s'abonne aux `device/`*`device_id`*`/data` MQTT rubriques où *`device_id`* figure l'identifiant de l'appareil qui a envoyé le message. Ces rubriques sont décrites dans un filtre de rubrique sous `device/+/data`, où la + est un caractère générique qui correspond à n'importe quelle chaîne de caractères comprise entre les deux barres obliques. Cette règle teste également la valeur du `temperature` champ dans la charge utile du message.

Lorsque la règle reçoit un message d'un sujet correspondant, elle prend le nom *`device_id`* du sujet, la `temperature` valeur de la charge utile du message, ajoute une valeur constante pour la limite qu'elle teste, puis envoie ces valeurs sous forme de JSON document à un sujet de SNS notification Amazon.

Par exemple, un MQTT message provenant du capteur météo numéro 32 utilise le `device/32/data` sujet et contient une charge utile du message qui ressemble à ceci :

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

L'instruction de requête de règle de la règle prend la `temperature` valeur de la charge utile du message, celle *`device_id`* du nom du sujet, et ajoute la `max_temperature` valeur constante pour envoyer une charge utile de message qui ressemble à ceci à la rubrique Amazon SNS :

```
{
```

```
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30
}
```

Pour créer une AWS IoT règle permettant de détecter une valeur de température supérieure à la limite et de créer les données à envoyer à la rubrique Amazon SNS

1. Ouvrez le hub de règles de la AWS IoT console.
2. S'il s'agit de votre première règle, choisissez Créer ou Créer une règle.
3. Dans Créer a règle: :
   a. Pour Name (Nom), entrez **`temp_limit_notify`**.

      N'oubliez pas qu'un nom de règle doit être unique dans votre région Compte AWS et qu'il ne doit pas comporter d'espaces. Nous avons utilisé un trait de soulignement dans ce nom pour séparer les mots du nom de la règle.

   b. Dans Description, décrivez la règle.

      Une description significative permet de se souvenir plus facilement du rôle de cette règle et de la raison pour laquelle vous l'avez créée. La description peut être aussi longue que nécessaire, donc soyez aussi détaillée que possible.
4. Dans l'déclararionde requête règle de Create a rule :
   a. Dans Utiliser SQL la version, sélectionnez 23/03/2016.
   b. Dans la zone d'édition de la instruction de requête de règle, entrez l'instruction :

      ```
      SELECT topic(2) as device_id,
          temperature as reported_temperature,
          30 as max_temperature
        FROM 'device/+/data'
        WHERE temperature > 30
      ```

      Cette instruction :

      - Écoute les MQTT messages dont le sujet correspond au filtre de `device/+/data` sujet et dont la `temperature` valeur est supérieure à 30.

- Sélectionne le deuxième élément dans la chaîne de l'objet et l'affecte au `device_id` champ.
- Sélectionne le `temperature` champ de valeur dans la charge utile du message et l'affecte au `reported_temperature` champ.
- Crée une valeur constante `30` pour représenter la valeur limite et l'affecte au `max_temperature` champ.

5. Pour ouvrir la liste des actions de règle pour cette règle, dans Définir une ou plusieurs actions, choisissez Ajouter une action.
6. Dans Sélectionnez une action, choisissez Envoyer un message sous forme de notification SNS push.
7. Pour ouvrir la page de configuration de l'action sélectionnée, en bas de la liste d'actions, choisissez Configurer l'action.
8. Sous Configurer les actions :

   a. Dans SNSTarget, choisissez Select, recherchez votre SNS sujet nommé high_temp_notice, puis sélectionnez Select.

   b. Dans Format du message, sélectionnez RAW.

   c. Dans Choisir ou créer un rôle pour accorder AWS IoT l'accès pour effectuer cette action, choisissez Créer un rôle.

   d. Dans Créer un nouveau rôle, dans Nom, entrez un nom unique pour le nouveau rôle. Dans le cadre de ce tutoriel, utilisez **`sns_rule_role`**.

   e. Sélectionnez Créer un rôle.

   Si vous répétez ce didacticiel ou si vous réutilisez un rôle existant, choisissez Mettre à jour le rôle avant de continuer. Cela met à jour le document de politique du rôle pour qu'il fonctionne avec la SNS cible.
9. Choisissez Ajouter une action et revenez à la page Créer une règle.

   Dans la vignette de la nouvelle action, sous Envoyer un message sous forme de notification SNS push, vous pouvez voir le SNS sujet concerné par votre règle.

   Il s'agit de la seule action de règle que vous ajouterez à cette règle.
10. Pour créer la règle et terminer cette étape, dans Créer une règle, faites défiler la page vers le bas et choisissez Créer une règle.

Étape 3 : tester la AWS IoT règle et la SNS notification Amazon

Pour tester votre nouvelle règle, vous allez utiliser le MQTT client pour publier les MQTT messages utilisés par cette règle et vous y abonner.

Ouvrez le MQTTclient dans la AWS IoT console dans une nouvelle fenêtre. Cela vous permettra de modifier la règle sans perdre la configuration de votre MQTT client. Si vous quittez le MQTT client pour accéder à une autre page de la console, celui-ci ne conservera aucun abonnement ni journal des messages.

Pour utiliser le MQTT client pour tester votre règle

1. Dans le MQTTclient de la AWS IoT console, abonnez-vous aux sujets d'entrée, dans ce cas,`device/+/data`.

   a. Dans le MQTT client, sous Abonnements, choisissez S'abonner à un sujet.
   b. Dans Sujet d'abonnement, entrez le objet du filtre de objet d'entrée,**`device/+/data`**.
   c. Conservez les valeurs par défaut des autres paramètres.
   d. Choisissez Subscribe to topic (S'abonner à la rubrique).

   Dans la colonne Abonnements, la section Publier dans un objet **`device/+/data`** apparaît.
2. Publiez un message sur le objet d'entrée avec un identifiant d'appareil spécifique, **`device/32/data`**. Vous ne pouvez pas publier dans MQTT des rubriques contenant des caractères génériques.

   a. Dans le MQTT client, sous Abonnements, choisissez Publier dans le sujet.
   b. Dans le champ Publier, entrez le nom de l'objet d'entrée, **`device/32/data`**.
   c. Copiez les exemples de données présentés ici et, dans la zone d'édition située sous le nom de l'objet, collez les exemples de données.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Choisissez Publier dans le sujet pour publier votre MQTT message.

3. Confirmez que le message texte a été envoyé.

   a. Dans le MQTT client, sous Abonnements, il y a un point vert à côté du sujet auquel vous vous êtes abonné précédemment.

      Le point vert indique qu'un ou plusieurs nouveaux messages ont été reçus depuis la dernière fois que vous les avez consultés.

   b. Sous Abonnements, choisissez device/+/data pour vérifier que la charge utile du message correspond à ce que vous venez de publier et ressemble à ceci :

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   c. Vérifiez le téléphone que vous avez utilisé pour vous abonner au SNS sujet et vérifiez que le contenu de la charge utile du message ressemble à ceci :

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

      Notez que la `device_id` valeur est une chaîne entre guillemets et que la `temperature` valeur est numérique. Cela est dû au fait que la `topic()` fonction a extrait la chaîne du nom de l'objet du message d'entrée tandis que la `temperature` valeur utilise la valeur numérique de la charge utile du message d'entrée.

      Si vous souhaitez transformer la `device_id` valeur en valeur numérique, remplacez-la `topic(2)` dans l'déclara l'instruction requête de règle par :

```
cast(topic(2) AS DECIMAL)
```

      Notez que la conversion de la `topic(2)` valeur numérique, `DECIMAL` ne fonctionnera que si cette partie de l'objet contient uniquement des caractères numériques.

4. Essayez d'envoyer un MQTT message dans lequel la température ne dépasse pas la limite.

a. Dans le MQTT client, sous Abonnements, choisissez Publier dans le sujet.

b. Dans le champ Publier, entrez le nom de l'objet d'entrée, **`device/33/data`**.

c. Copiez les exemples de données présentés ici et, dans la zone d'édition située sous le nom de l'objet, collez les exemples de données.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Pour envoyer votre MQTT message, choisissez Publier dans le sujet.

Vous devriez voir le message que vous avez envoyé dans l'**`device/+/data`**abonnement. Toutefois, étant donné que la valeur de température est inférieure à la température maximale indiquée dans l'déclararionde requête de règle, vous ne devriez pas recevoir de message texte.

Si vous ne voyez pas le comportement correct, consultez les conseils de dépannage.

Résolution des problèmes liés à votre règle de SNS message

Voici quelques points à vérifier, au cas où vous n'obtiendriez pas les résultats escomptés.

- Vous avez reçu une bannière d'erreur

  Si une erreur est apparue lorsque vous avez publié le message d'entrée, corrigez-la d'abord. Les étapes suivantes peuvent vous aider à corriger cette erreur.

- Vous ne voyez pas le message d'entrée dans le MQTT client

  Chaque fois que vous publiez votre message d'entrée dans le `device/22/data` sujet, ce message doit apparaître dans le MQTT client, si vous vous êtes abonné au filtre de `device/+/data` sujet comme décrit dans la procédure.

À savoir

- Vérifiez le filtre de objets auquel vous vous êtes abonné

  Si vous vous êtes abonné au objet du message d'entrée comme décrit dans la procédure, vous devriez voir une copie du message d'entrée chaque fois que vous le publiez.

  Si le message ne s'affiche pas, vérifiez le nom de l'objet auquel vous vous êtes abonné et comparez-le au objet dans lequel vous avez publié. Les noms des objets distinguent les majuscules et minuscules et le objet auquel vous vous êtes abonné doit être identique au objet dans lequel vous avez publié la charge utile du message.

- Vérifiez la fonction de publication des messages

  Dans le MQTT client, sous Abonnements, choisissez device/+/data, vérifiez le sujet du message de publication, puis choisissez Publier dans le sujet. La charge utile du message figurant dans la zone d'édition située sous le objet devrait apparaître dans la liste des messages.

- Vous ne recevez pas de SMS message

  Pour que votre règle fonctionne, elle doit disposer de la politique appropriée qui l'autorise à recevoir un message et à envoyer une SNS notification, et elle doit recevoir le message.

  À savoir

  - Vérifiez le Région AWS nom de votre MQTT client et la règle que vous avez créée

    La console dans laquelle vous exécutez le MQTT client doit se trouver dans la même AWS région que la règle que vous avez créée.

  - Vérifiez que la valeur de température dans la charge utile du message dépasse le seuil de test

    Si la valeur de température est inférieure ou égale à 30, telle que définie dans l'déclararionde requête de règle, la règle n'exécutera aucune de ses actions.

  - Vérifiez le objet du message d'entrée dans la déclaration de requête de règle

    Pour que la règle fonctionne, elle doit recevoir un message dont le nom de rubrique correspond au filtre de rubrique figurant dans la FROM clause de l'instruction de requête de règle.

    Vérifiez l'orthographe du filtre de rubrique dans l'instruction de requête de règle avec celle du sujet dans le MQTT client. Les noms de objets distinguent les majuscules et minuscules et le

objet du message doit correspondre au filtre de objet indiqué dans l'déclararionde requête de règle.

- Vérifiez le contenu de la charge utile des messages d'entrée

  Pour que la règle fonctionne, elle doit trouver le champ de données dans la charge utile du message déclarée dans l'SELECTinstruction.

  Vérifiez l'orthographe du `temperature` champ dans l'instruction de requête de règle avec celle de la charge utile du message dans le MQTT client. Les noms de champs distinguent les majuscules et minuscules et le `temperature` champ de l'déclararionde requête de règle doit être identique au `temperature` champ de la charge du message.

  Assurez-vous que le JSON document contenu dans la charge utile du message est correctement formaté. S'il JSON contient des erreurs, telles qu'une virgule manquante, la règle ne pourra pas le lire.

- Vérifiez le objet du message republié dans l'action de la règle

  Le sujet auquel l'action Republier la règle publie le nouveau message doit correspondre au sujet auquel vous vous êtes abonné dans le MQTT client.

  Ouvrez la règle que vous avez créée dans la console et vérifiez le objet dans lequel l'action de règle republiera le message.

- Vérifiez le rôle utilisé par la règle

  L'action de règle doit être autorisée à recevoir le objet d'origine et à publier le nouveau objet.

  Les politiques qui autorisent la règle à recevoir les données des messages et à les republier sont spécifiques aux objets utilisés. Si vous modifiez le objet utilisé pour republier les données du message, vous devez mettre à jour le rôle de l'action de règle afin de mettre à jour sa politique afin qu'elle corresponde au objet actuel.

  Si vous pensez que c'est le problème, modifiez l'action Republier la règle et créez un nouveau rôle. Les nouveaux rôles créés par l'action de règle reçoivent les autorisations nécessaires pour effectuer ces actions.

Étape 4 : examen des résultats et des étapes suivantes

Dans ce tutoriel :

- Vous avez créé et testé un sujet de SNS notification et un abonnement Amazon.
- Vous avez utilisé une SQL requête et des fonctions simples dans une instruction de requête de règle pour créer un nouveau message pour votre notification.
- Vous avez créé une AWS IoT règle pour envoyer une SNS notification Amazon en utilisant la charge utile de vos messages personnalisés.
- Vous avez utilisé le MQTT client pour tester votre AWS IoT règle.

Étapes suivantes

Après avoir envoyé quelques SMS avec cette règle, essayez de l'utiliser pour voir comment la modification de certains aspects du didacticiel affecte le message et le moment où il est envoyé. Voici quelques idées pour vous aider à démarrer.

- Changez le *`device_id`* dans le sujet du message d'entrée et observez l'effet dans le contenu du message texte.
- Modifiez les champs sélectionnés dans l'déclararionde requête de règle et observez l'effet dans le contenu du message texte.
- Modifiez le test dans l'déclararionde requête de règle pour tester une température minimale au lieu d'une température maximale. N'oubliez pas de changer le nom de `max_temperature` !
- Ajoutez une action de règle de republication pour envoyer un MQTT message lorsqu'une SNS notification est envoyée.
- Essayez le prochain didacticiel de cette série et découvrez comment Tutoriel : Stockage des données de l'appareil dans une table DynamoDB .

Tutoriel : Stockage des données de l'appareil dans une table DynamoDB

Ce didacticiel explique comment créer une AWS IoT règle qui envoie des données de message à une table DynamoDB.

Dans ce didacticiel, vous allez créer une règle qui envoie les données des messages d'un capteur météo imaginaire vers une table DynamoDB. La règle met en forme les données de nombreux capteurs météorologiques de manière à ce qu'elles puissent être ajoutées à une seule table de base de données.

Ce que vous allez apprendre dans ce didacticiel

- Pour créer une table DynamoDB
- Comment envoyer des données de message à une table DynamoDB à partir d'une règle AWS IoT
- Comment utiliser des modèles de substitution dans une AWS IoT règle
- Comment utiliser des SQL requêtes et des fonctions simples dans une instruction de requête de règle
- Comment utiliser le MQTT client pour tester une AWS IoT règle

Ce didacticiel vous prendra environ 30 minutes.

Dans ce tutoriel, vous allez :

- Dans l'étape 1 de ce didacticiel, vous allez créer une table dans DynamoDB en utilisant l'.
- Étape 2 : créer une AWS IoT règle pour envoyer des données à la table DynamoDB
- Étape 3 : tester la AWS IoT règle et la table DynamoDB
- Étape 4 : examen des résultats et des étapes suivantes

Avant de commencer ce didacticiel, assurez-vous de disposer des éléments suivants :

- Configurez Compte AWS

  Vous aurez besoin de votre AWS IoT console Compte AWS et de votre console pour terminer ce didacticiel.
- Révisé Afficher les messages MQTT avec le client AWS IoT MQTT

  Assurez-vous que vous pouvez utiliser le MQTT client pour vous abonner à un sujet et le publier. Vous allez utiliser le MQTT client pour tester votre nouvelle règle dans le cadre de cette procédure.
- J'ai consulté la présentation d'Amazon DynamoDB

  Si vous n'avez jamais utilisé DynamoDB auparavant, consultez Mise en route with DynamoDB pour vous familiariser avec les concepts et opérations de DynamoDB.

Dans l'étape 1 de ce didacticiel, vous allez créer une table dans DynamoDB en utilisant l'.

Dans ce didacticiel, vous allez créer une table DynamoDB avec les attributs suivants pour enregistrer les données des capteurs météorologiques imaginaires :

- `sample_time`est une clé primaire et décrit l'heure à laquelle l'échantillon a été enregistré.
- `device_id`est une clé de tri et décrit le périphérique qui a fourni l'échantillon
- `device_data`correspond aux données reçues de l'appareil et formatées par l'déclararionde requête de règle

Pour créer la table DynamoDB pour ce didacticiel

1. Ouvrez la console DynamoDB, puis choisissez Créer une un tableau.
2. Dans Créer une table :
   a. Saisissez un nom de table dans la zone Nom de la table.
   b. Dans Clé de partition, entrez **`sample_time`**, puis dans la liste d'options à côté du champ, sélectionnez **`Number`**.
   c. Dans Touche de tri, entrez **`device_id`**, et dans la liste d'options à côté du champ, sélectionnez **`Number`**.
   d. En bas de la page, choisissez Create .

Vous le définirez `device_data` ultérieurement, lorsque vous configurerez l'action de règle DynamoDB.

Étape 2 : créer une AWS IoT règle pour envoyer des données à la table DynamoDB

Au cours de cette étape, vous allez utiliser l'déclararionde requête de règle pour formater les données provenant des capteurs météorologiques imaginaires afin de les écrire dans la table de base de données.

Voici un exemple de charge utile de messages reçue d'un capteur météo :

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Pour l'entrée de base de données, vous allez utiliser l'déclararionrule query pour aplatir la structure de la charge utile du message de manière à ce qu'elle ressemble à ceci :

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind_velocity": 22,
  "wind_bearing": 255
}
```

Dans cette règle, vous utiliserez également deux Modèles de substitution. Les modèles de substitution sont des expressions qui vous permettent d'insérer des valeurs dynamiques à partir de fonctions et de données de message.

Pour créer la AWS IoT règle permettant d'envoyer des données à la table DynamoDB

1. Ouvrez les Règles du hub de la console AWS IoT. Vous pouvez également ouvrir la AWS IoT page d'accueil dans le AWS Management Console et accéder à Routage des messages > Règles.
2. Pour commencer à créer votre nouvelle règle dans Règles, choisissez Créer une règle.
3. Dans Propriétés de la règle :

   a. Sous Nom du rôle, entrez **`wx_data_ddb`**.

      N'oubliez pas qu'un nom de règle doit être unique dans votre région Compte AWS et qu'il ne doit pas comporter d'espaces. Nous avons utilisé un trait de soulignement dans ce nom pour séparer les deux mots du nom de la règle.

   b. Dans Description de la règle, décrivez la règle.

      Une description significative permet de se souvenir plus facilement du rôle de cette règle et de la raison pour laquelle vous l'avez créée. La description peut être aussi longue que nécessaire, donc soyez aussi détaillée que possible.
4. Choisissez Next (Suivant) pour continuer.
5. Dans SQLune déclaration :

   a. Dans SQLla version, sélectionnez**`2016-03-23`**.

   b. Dans la zone d'édition du SQLrelevé, entrez le relevé :

```
SELECT temperature, humidity, barometer,
  wind.velocity as wind_velocity,
  wind.bearing as wind_bearing,
FROM 'device/+/data'
```

Cette instruction :

- Écoute les MQTT messages dont le sujet correspond au filtre de `device/+/data` sujet.
- Formate les éléments de l'`wind`attribut en tant qu'attributs individuels.
- Transmet les `temperature`, `humidity`, et `barometer`, attributs et tels quels.

6. Choisissez Next (Suivant) pour continuer.
7. Dans Rule actions :

   a. Pour ouvrir la liste des actions de règle pour cette règle, dans Action 1, sélectionnez**DynamoDB**.

   > **Note**
   >
   > Assurez-vous de choisir DynamoDB et non ynamoDBv D2 comme action de règle.

   b. Dans Nom de la table, choisissez le nom de la table DynamoDB que vous avez créée lors d'une étape précédente :. **`wx_data`**

   Les champs Type de clé de partition et Type de clé de tri sont remplis avec les valeurs de votre tableau DynamoDB.

   c. Dans Clé de partition, entrer **`sample_time`**.

   d. Dans Valeur de la clé de partition, tapez **`${timestamp()}`**.

   Il s'agit de la première des Modèles de substitution que vous utiliserez dans cette règle. Au lieu d'utiliser une valeur provenant de la charge utile du message, il utilisera la valeur renvoyée par la fonction d'horodatage. Pour en savoir plus, consultez timestamp dans le AWS IoT Core Guide du développeur.

   e. Dans clé de tri, entrez **`device_id`**.

   f. Dans Valeur de la clé de tri, entrez **`${cast(topic(2) AS DECIMAL)}`**.

Il s'agit de la deuxième des Modèles de substitution que vous utiliserez dans cette règle. Il insère la valeur du deuxième élément dans le nom du sujet, qui est l'ID de l'appareil, après l'avoir converti en une DECIMAL valeur correspondant au format numérique de la clé. Pour de plus amples informations, veuillez consulter Rubriques dans le AWS IoT Core Guide du développeur. Ou pour en savoir plus sur le casting, consultez le AWS IoT Core guide du développeur.

g. Dans Write message data to this column (Écrire des données de message dans cette colonne), entrez **`device_data`**.

   Cela créera la `device_data` colonne dans la table DynamoDB.

h. Laissez le champ Operation (Opération) vide.

i. Dans IAMrôle, choisissez Créer un nouveau rôle.

j. Dans la boîte de dialogue Créer un rôle, pour Nom du rôle, entrez wx_ddb_role. Ce nouveau rôle contiendra automatiquement une politique avec le préfixe « aws-iot-rule » qui permettra à la **`wx_data_ddb`** règle d'envoyer des données à la table **`wx_data`** DynamoDB que vous avez créée.

k. Dans IAMle rôle, choisissez**`wx_ddb_role`**.

l. Au bas de la page, sélectionnez Next.

8. Au bas de la page Réviser et créer, choisissez Créer pour créer la règle.

Étape 3 : tester la AWS IoT règle et la table DynamoDB

Pour tester la nouvelle règle, vous allez utiliser le MQTT client pour publier les MQTT messages utilisés dans ce test et vous y abonner.

Ouvrez le MQTTclient dans la AWS IoT console dans une nouvelle fenêtre. Cela vous permettra de modifier la règle sans perdre la configuration de votre MQTT client. Le MQTT client ne conserve aucun abonnement ou journal des messages si vous le quittez pour accéder à une autre page de la console. Vous devez également ouvrir une fenêtre de console distincte sur le hub de tables DynamoDB de AWS IoT la console pour afficher les nouvelles entrées envoyées par votre règle.

Pour utiliser le MQTT client pour tester votre règle

1. Dans le MQTTclient de la AWS IoT console, abonnez-vous au sujet d'entrée,`device/+/data`.

   a. Dans le MQTT client, choisissez S'abonner à un sujet.

b. Pour le filtre de objet, entrez l'objet du filtre de l'objet d'entrée, **`device/+/data`**.

c. Choisissez Souscrire.

2. Maintenant, publiez un message sur le objet d'entrée avec un identifiant d'appareil spécifique, **`device/22/data`**. Vous ne pouvez pas publier dans MQTT des rubriques contenant des caractères génériques.

a. Dans le MQTT client, choisissez Publier dans un sujet.

b. Dans Nom de la rubrique, attribuez un nom à la rubrique **`device/22/data`**.

c. Pour la charge du message, entrez les exemples de données suivants.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Pour publier le MQTT message, choisissez Publier.

e. Maintenant, dans le MQTT client, choisissez S'abonner à un sujet. Dans la colonne S'abonner, choisissez l'**`device/+/data`**abonnement. Vérifiez que les exemples de données de l'étape précédente y figurent.

3. Vérifiez la ligne de la table DynamoDB créée par votre règle.

a. Dans le hub de tables DynamoDB de AWS IoT la console, choisissez wx_data, puis l'onglet Éléments.

Si vous êtes déjà dans l'onglet Éléments savoir, vous devrez peut-être actualiser l'affichage en choisissant l'icône d'actualisation dans le coin supérieur droit de l'en-tête du tableau.

b. Notez que les valeurs sample_time de la table sont des liens et ouvrez-en un. Si vous venez d'envoyer votre premier message, ce sera le seul de la liste.

Ce lien affiche toutes les données de cette ligne du tableau.

c. Développez l'entrée device_data pour voir les données résultant de l'déclararionde requête de règle.

d. Explorez les différentes représentations des données disponibles dans cet affichage. Vous pouvez également modifier les données de cet affichage.

e. Après avoir passé en revue cette ligne de données, pour enregistrer les modifications que vous avez apportées, choisissez Enregistrer, ou pour quitter sans enregistrer les modifications, choisissez Annuler.

Si vous ne voyez pas le comportement correct, consultez les conseils de dépannage.

Résolution des problèmes de votre règle DynamoDB

Voici quelques points à vérifier au cas où vous n'obtiendriez pas les résultats escomptés.

- Vous avez reçu une bannière d'erreur

  Si une erreur est apparue lorsque vous avez publié le message d'entrée, corrigez-la d'abord. Les étapes suivantes peuvent vous aider à corriger cette erreur.
- Vous ne voyez pas le message d'entrée dans le MQTT client

  Chaque fois que vous publiez votre message d'entrée dans le `device/22/data` sujet, ce message doit apparaître dans le MQTT client si vous vous êtes abonné au filtre de `device/+/data` sujet comme décrit dans la procédure.

  À savoir

  - Vérifiez le filtre de objets auquel vous vous êtes abonné

    Si vous vous êtes abonné au objet du message d'entrée comme décrit dans la procédure, vous devriez voir une copie du message d'entrée chaque fois que vous le publiez.

    Si le message ne s'affiche pas, vérifiez le nom de l'objet auquel vous vous êtes abonné et comparez-le au objet dans lequel vous avez publié. Les noms des objets distinguent les majuscules et minuscules et le objet auquel vous vous êtes abonné doit être identique au objet dans lequel vous avez publié la charge utile du message.
  - Vérifiez la fonction de publication des messages

    Dans le MQTT client, sous Abonnements, choisissez device/+/data, vérifiez le sujet du message de publication, puis choisissez Publier dans le sujet. La charge utile du message figurant dans la zone d'édition située sous le objet devrait apparaître dans la liste des messages.
- Vous ne voyez pas vos données dans la table DynamoDB

La première chose à faire est d'actualiser l'affichage en choisissant l'icône d'actualisation dans le coin supérieur droit de l'en-tête du tableau. Si les données que vous recherchez ne s'affichent pas, vérifiez les points suivants.

À savoir

- Vérifiez le Région AWS nom de votre MQTT client et la règle que vous avez créée

  La console dans laquelle vous exécutez le MQTT client doit se trouver dans la même AWS région que la règle que vous avez créée.
- Vérifiez le objet du message d'entrée dans la déclaration de requête de règle

  Pour que la règle fonctionne, elle doit recevoir un message dont le nom de rubrique correspond au filtre de rubrique figurant dans la FROM clause de l'instruction de requête de règle.

  Vérifiez l'orthographe du filtre de rubrique dans l'instruction de requête de règle avec celle du sujet dans le MQTT client. Les noms de objets distinguent les majuscules et minuscules et le objet du message doit correspondre au filtre de objet indiqué dans l'déclararionde requête de règle.
- Vérifiez le contenu de la charge utile des messages d'entrée

  Pour que la règle fonctionne, elle doit trouver le champ de données dans la charge utile du message déclarée dans l'SELECTinstruction.

  Vérifiez l'orthographe du `temperature` champ dans l'instruction de requête de règle avec celle de la charge utile du message dans le MQTT client. Les noms de champs distinguent les majuscules et minuscules et le `temperature` champ de l'déclararionde requête de règle doit être identique au `temperature` champ de la charge du message.

  Assurez-vous que le JSON document contenu dans la charge utile du message est correctement formaté. S'il JSON contient des erreurs, telles qu'une virgule manquante, la règle ne pourra pas le lire.
- Vérifiez les noms de clé et de champ utilisés dans l'action de la règle

  Les noms de champs utilisés dans la règle de rubrique doivent correspondre à ceux trouvés dans la JSON charge utile du message publié.

  Ouvrez la règle que vous avez créée dans la console et vérifiez les noms des champs dans la configuration de l'action des règles avec ceux utilisés dans le MQTT client.

- Vérifiez le rôle utilisé par la règle

  L'action de règle doit être autorisée à recevoir le objet d'origine et à publier le nouveau objet.

  Les politiques qui autorisent la règle à recevoir des données de message et à mettre à jour la table DynamoDB sont spécifiques aux rubriques utilisées. Si vous modifiez le nom de rubrique ou de table DynamoDB utilisé par la règle, vous devez mettre à jour le rôle de l'action de règle afin de mettre à jour sa politique en conséquence.

  Si vous pensez que c'est le problème, modifiez l'action de la règle et créez un nouveau rôle. Les nouveaux rôles créés par l'action de règle reçoivent les autorisations nécessaires pour effectuer ces actions.

Étape 4 : examen des résultats et des étapes suivantes

Après avoir envoyé quelques messages à la table DynamoDB avec cette règle, essayez de l'utiliser pour voir comment la modification de certains aspects du didacticiel affecte les données écrites dans la table. Voici quelques idées pour vous aider à démarrer.

- Changez le *`device_id`* dans le sujet du message d'entrée et observez l'effet sur les données. Vous pouvez l'utiliser pour simuler la réception de données provenant de plusieurs capteurs météorologiques.
- Modifiez les champs sélectionnés dans l'déclararionde requête de règle et observez l'effet sur les données. Vous pouvez l'utiliser pour filtrer les données stockées dans la table.
- Ajoutez une action de règle de republication pour envoyer un MQTT message pour chaque ligne ajoutée au tableau. Vous pouvez l'utiliser pour le débogage.

Après avoir terminé ce didacticiel, jetez un œil the section called “Formatage d'une notification à l'aide d'une AWS Lambda fonction”.

Tutoriel : Formatage d'une notification à l'aide d'une AWS Lambda fonction

Ce didacticiel explique comment envoyer des données de MQTT message à une AWS Lambda action pour le formatage et l'envoi à un autre AWS service. Dans ce didacticiel, l' AWS Lambda action utilise le AWS SDK pour envoyer le message formaté au SNS sujet Amazon que vous avez créé dans le didacticiel sur la procédure à the section called “Envoi d'une SNS notification Amazon” suivre.

Dans le didacticiel expliquant comment procéderthe section called “Envoi d'une SNS notification Amazon”, le JSON document issu de l'instruction de requête de la règle a été envoyé dans le corps du message texte. Le résultat a été un message texte qui ressemblait à cet exemple :

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

Dans ce didacticiel, vous allez utiliser une action de AWS Lambda règle pour appeler une AWS Lambda fonction qui met en forme les données de l'instruction de requête de règle dans un format plus convivial, comme dans cet exemple :

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

La AWS Lambda fonction que vous allez créer dans ce didacticiel formate la chaîne du message en utilisant les données de l'instruction de requête de règle et appelle la fonction de SNSpublication du AWS SDK pour créer la notification.

Ce que vous allez apprendre dans ce didacticiel

- Comment créer et tester une AWS Lambda fonction
- Comment utiliser la AWS Lambda fonction AWS SDK in an pour publier une SNS notification Amazon
- Comment utiliser des SQL requêtes et des fonctions simples dans une instruction de requête de règle
- Comment utiliser le MQTT client pour tester une AWS IoT règle

Ce didacticiel vous prendra environ 45 minutes.

Dans ce tutoriel, vous allez :

- Étape 1 : créer une AWS Lambda fonction qui envoie un message texte
- Étape 2 : Création d'une AWS IoT règle avec une action de AWS Lambda règle
- Étape 3 : tester la AWS IoTAWS Lambda règle et son action
- Étape 4 : examen des résultats et des étapes suivantes

Avant de commencer ce didacticiel, assurez-vous de disposer des éléments suivants :

- Configurez Compte AWS

Vous aurez besoin de votre AWS IoT console Compte AWS et de votre console pour terminer ce didacticiel.

- Révisé Afficher les messages MQTT avec le client AWS IoT MQTT

  Assurez-vous que vous pouvez utiliser le MQTT client pour vous abonner à un sujet et le publier. Vous allez utiliser le MQTT client pour tester votre nouvelle règle dans le cadre de cette procédure.

- Vous avez terminé les autres didacticiels sur les règles de cette section

  Ce didacticiel nécessite le sujet de SNS notification que vous avez créé dans le didacticiel pour savoir comment procéderthe section called “Envoi d'une SNS notification Amazon”. Cela suppose également que vous avez suivi les autres didacticiels relatifs aux règles de cette section.

- J'ai revu l'AWS Lambda aperçu

  Si vous ne l'avez jamais utilisé AWS Lambda auparavant, consultez le AWS Lambdaguide Getting started with Lambda pour en apprendre les termes et les concepts.

Étape 1 : créer une AWS Lambda fonction qui envoie un message texte

La AWS Lambda fonction de ce didacticiel reçoit le résultat de l'instruction de requête de règle, insère les éléments dans une chaîne de texte et envoie la chaîne résultante à Amazon SNS sous forme de message dans une notification.

Contrairement au didacticiel expliquant commentthe section called “Envoi d'une SNS notification Amazon”, qui utilisait une action de AWS IoT règle pour envoyer la notification, ce didacticiel envoie la notification à partir de la fonction Lambda en utilisant une fonction du. AWS SDK Le sujet des SNS notifications Amazon utilisé dans ce didacticiel est toutefois le même que celui que vous avez utilisé dans le didacticiel expliquant comment procéderthe section called “Envoi d'une SNS notification Amazon”.

Pour créer une AWS Lambda fonction qui envoie un message texte

1. Créez une nouvelle AWS Lambda fonction.

   a. Dans la AWS Lambda console, choisissez Créer une fonction.
   b. Dans Créer une fonction, sélectionnez Utiliser un plan.

      Recherchez et sélectionnez le **`hello-world-python`** plan, puis choisissez Configurer.
   c. Dans Informations de base :

i. Dans Nom de la fonction, entrez le nom de votre fonction, par exemple **`format-high-temp-notification`**.

ii. Dans Rôle d'exécution, choisissez Créer un nouveau rôle à partir de modèles de AWS politique.

iii. Dans Role name, entrez un nom pour le nouveau rôle.

iv. Dans Modèles de politiques (facultatif), recherchez et sélectionnez Amazon SNS Publish Policy.

v. Sélectionnez Create function (Créer une fonction).

2. Modifiez le code du plan pour le formater et envoyer une SNS notification Amazon.

a. Après avoir créé votre fonction, vous devriez voir la page de format-high-temp-notificationdétails. Si ce n'est pas le cas, ouvrez-le depuis la page des Lambdafonctions.

b. Sur la page de format-high-temp-notificationdétails, choisissez l'onglet Configuration et faites défiler l'écran jusqu'au panneau des codes de fonction.

c. Dans la fenêtre Code de fonction, dans le volet Environnement, choisissez le fichier Python, `lambda_function.py`.

d. Dans la fenêtre Code de fonction, supprimez tout le code du programme d'origine du plan et remplacez-le par ce code.

```
import boto3
#
#   expects event parameter to contain:
#   {
#       "device_id": "32",
#       "reported_temperature": 38,
#       "max_temperature": 30,
#       "notify_topic_arn": "arn:aws:sns:us-
east-1:57EXAMPLE833:high_temp_notice"
#   }
#
#   sends a plain text string to be used in a text message
#
#      "Device {0} reports a temperature of {1}, which exceeds the limit of
 {2}."
#
#   where:
#       {0} is the device_id value
#       {1} is the reported_temperature value
```

```
#       {2} is the max_temperature value
#
def lambda_handler(event, context):

    # Create an SNS client to send notification
    sns = boto3.client('sns')

    # Format text message from data
    message_text = "Device {0} reports a temperature of {1}, which exceeds the
 limit of {2}.".format(
            str(event['device_id']),
            str(event['reported_temperature']),
            str(event['max_temperature'])
        )

    # Publish the formatted message
    response = sns.publish(
            TopicArn = event['notify_topic_arn'],
            Message = message_text
        )

    return response
```

e. Choisissez Deploy (Déployer).

3. Dans une nouvelle fenêtre, recherchez le nom de la ressource Amazon (ARN) de votre SNS rubrique Amazon dans le didacticiel expliquant comment procéderthe section called “Envoi d'une SNS notification Amazon”.

   a. Dans une nouvelle fenêtre, ouvrez la page Sujets de la SNS console Amazon.

   b. Sur la page Rubriques, recherchez la rubrique de notification high_temp_notice dans la liste des rubriques Amazon. SNS

   c. Recherchez la rubrique ARNde notification high_temp_notice à utiliser à l'étape suivante.

4. Créez un scénario de test pour votre fonction Lambda.

   a. Sur la page Fonctions Lambda de la console, sur la page de format-high-temp-notificationdétails, choisissez Sélectionner un événement de test dans le coin supérieur droit de la page (même s'il semble désactivé), puis sélectionnez Configurer les événements de test.

   b. Dans Configure test event (Configurer un événement de test) choisissez Select a test event (Sélectionner un événement de test).

c. Dans Event name (Nom de l'événement), saisissez **`SampleRuleOutput`**.

d. Dans l'JSONéditeur situé sous Nom de l'événement, collez cet exemple de JSON document. Voici un exemple de ce que votre AWS IoT règle enverra à la fonction Lambda.

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

e. Reportez-vous à la fenêtre contenant le sujet ARNde notification high_temp_notice et copiez la valeur. ARN

f. Remplacez la `notify_topic_arn` valeur dans l'JSONéditeur par celle ARN de votre sujet de notification.

Gardez cette fenêtre ouverte afin de pouvoir réutiliser cette ARN valeur lors de la création de la AWS IoT règle.

g. Sélectionnez Create (Créer).

5. Testez la fonction avec des exemples de données.

a. Sur la page de format-high-temp-notificationdétails, dans le coin supérieur droit de la page, vérifiez que cela SampleRuleOutputapparaît à côté du bouton Test. Si ce n'est pas le cas, sélectionnez-le dans la liste des événements de test disponibles.

b. Pour envoyer le message de sortie de l'exemple de règle à votre fonction, choisissez Test.

Si la fonction et la notification ont toutes deux fonctionné, vous recevrez un message texte sur le téléphone abonné à la notification.

Si vous n'avez pas reçu de SMS au téléphone, vérifiez le résultat de l'opération. Dans le panneau Code de fonction, dans l'onglet Résultat de l'exécution, passez en revue la réponse pour détecter les erreurs qui se sont produites. Ne passez pas à l'étape suivante tant que votre fonction n'a pas pu envoyer la notification à votre téléphone.

Étape 2 : Création d'une AWS IoT règle avec une action de AWS Lambda règle

Au cours de cette étape, vous allez utiliser l'déclararionde requête de règle pour formater les données provenant du capteur météo imaginaire à envoyer à une fonction Lambda, qui formatera et enverra un message texte.

Voici un exemple de charge utile des messages reçus des appareils météorologiques :

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

Dans cette règle, vous allez utiliser l'déclararionrule query pour créer une charge utile de message pour la fonction Lambda qui ressemble à ceci :

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

Il contient toutes les informations dont la fonction Lambda a besoin pour formater et envoyer le message texte correct.

Pour créer la AWS IoT règle permettant d'appeler une fonction Lambda

1. Ouvrez le hub de règles de la AWS IoT console.
2. Pour commencer à créer votre nouvelle règle dans Règles, choisissez Créer.
3. Dans la partie supérieure de Créer une règle :
   a. Dans Nom, entrez le nom de la règle, **`wx_friendly_text`**.

N'oubliez pas qu'un nom de règle doit être unique dans votre région Compte AWS et qu'il ne doit pas comporter d'espaces. Nous avons utilisé un trait de soulignement dans ce nom pour séparer les deux mots du nom de la règle.

b. Dans Description, décrivez la règle.

Une description significative permet de se souvenir plus facilement du rôle de cette règle et de la raison pour laquelle vous l'avez créée. La description peut être aussi longue que nécessaire, donc soyez aussi détaillée que possible.

4. Dans l'déclararionde requête règle de Create a rule :

a. Dans Utiliser SQL la version, sélectionnez**2016-03-23**.

b. Dans la zone d'édition de la instruction de requête de règle, entrez l'instruction :

```
SELECT
  cast(topic(2) AS DECIMAL) as device_id,
  temperature as reported_temperature,
  30 as max_temperature,
  'arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice' as notify_topic_arn
FROM 'device/+/data' WHERE temperature > 30
```

Cette instruction :

- Écoute les MQTT messages dont le sujet correspond au filtre de `device/+/data` sujet et dont la `temperature` valeur est supérieure à 30.
- Sélectionne le deuxième élément dans la chaîne de l'objet, le convertit en nombre décimal, puis l'affecte au `device_id` champ.
- Sélectionne la valeur du `temperature` champ dans la charge utile du message et l'affecte au `reported_temperature` champ.
- Crée une valeur constante, `30`, pour représenter la valeur limite et l'affecte au `max_temperature` champ.
- Crée une valeur constante pour le `notify_topic_arn` champ.

c. Reportez-vous à la fenêtre contenant le sujet ARNde notification high_temp_notice et copiez la valeur. ARN

d. Remplacez la ARN valeur (*arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice*) dans l'éditeur d'instructions de requête de règles avec le sujet ARN de votre notification.

5. Dans Définir une ou plusieurs actions :

   a. Pour ouvrir la liste des actions de règle pour cette règle, choisissez Ajouter une action.

   b. Dans Sélectionner une action, choisissez Envoyer un message à une fonction Lambda.

   c. Pour ouvrir la page de configuration de l'action sélectionnée, en bas de la liste d'actions, choisissez Configurer l'action.

6. Sous Configurer les actions :

   a. Dans Nom de la fonction, choisissez Sélectionner.

   b. Choisissez format-high-temp-notification.

   c. Au bas de Configurer l'action, choisissez Ajouter une action.

   d. Pour créer la règle, en bas de Créer une règle, sélectionnez Créer une règle.

Étape 3 : tester la AWS IoTAWS Lambda règle et son action

Pour tester votre nouvelle règle, vous allez utiliser le MQTT client pour publier les MQTT messages utilisés par cette règle et vous y abonner.

Ouvrez le MQTTclient dans la AWS IoT console dans une nouvelle fenêtre. Vous pouvez désormais modifier la règle sans perdre la configuration de votre MQTT client. Si vous quittez le MQTT client pour accéder à une autre page de la console, vous perdrez vos abonnements ou vos journaux de messages.

Pour utiliser le MQTT client pour tester votre règle

1. Dans le MQTTclient de la AWS IoT console, abonnez-vous aux sujets d'entrée, dans ce cas,`device/+/data`.

   a. Dans le MQTT client, sous Abonnements, choisissez S'abonner à un sujet.

   b. Dans Sujet d'abonnement, entrez le objet du filtre de objet d'entrée,**`device/+/data`**.

   c. Conservez les valeurs par défaut des autres paramètres.

   d. Choisissez Subscribe to topic (S'abonner à la rubrique).

   Dans la colonne Abonnements, la section Publier dans un objet **`device/+/data`** apparaît.

2. Publiez un message sur le objet d'entrée avec un identifiant d'appareil spécifique, **`device/32/data`**. Vous ne pouvez pas publier dans MQTT des rubriques contenant des caractères génériques.

a. Dans le MQTT client, sous Abonnements, choisissez Publier dans le sujet.

b. Dans le champ Publier, entrez le nom de l'objet d'entrée, **`device/32/data`**.

c. Copiez les exemples de données présentés ici et, dans la zone d'édition située sous le nom de l'objet, collez les exemples de données.

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. Pour publier votre MQTT message, choisissez Publier dans le sujet.

3. Confirmez que le message texte a été envoyé.

a. Dans le MQTT client, sous Abonnements, il y a un point vert à côté du sujet auquel vous vous êtes abonné précédemment.

Le point vert indique qu'un ou plusieurs nouveaux messages ont été reçus depuis la dernière fois que vous les avez consultés.

b. Sous Abonnements, choisissez device/+/data pour vérifier que la charge utile du message correspond à ce que vous venez de publier et ressemble à ceci :

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. Vérifiez le téléphone que vous avez utilisé pour vous abonner au SNS sujet et vérifiez que le contenu de la charge utile du message ressemble à ceci :

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

Si vous modifiez l'élément ID de objet dans le objet du message, n'oubliez pas que la conversion de la `topic(2)` valeur en valeur numérique ne fonctionnera que si cet élément de l'objet du message contient uniquement des caractères numériques.

4. Essayez d'envoyer un MQTT message dans lequel la température ne dépasse pas la limite.

   a. Dans le MQTT client, sous Abonnements, choisissez Publier dans le sujet.

   b. Dans le champ Publier, entrez le nom de l'objet d'entrée, **`device/33/data`**.

   c. Copiez les exemples de données présentés ici et, dans la zone d'édition située sous le nom de l'objet, collez les exemples de données.

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. Pour envoyer votre MQTT message, choisissez Publier dans le sujet.

   Vous devriez voir le message que vous avez envoyé dans l'**`device/+/data`**abonnement ; toutefois, comme la valeur de température est inférieure à la température maximale indiquée dans l'énoncé de la requête de règle, vous ne devriez pas recevoir de message texte.

   Si vous ne voyez pas le comportement correct, consultez les conseils de dépannage.

Résolution des problèmes liés à votre AWS Lambda règle et à votre notification

Voici quelques points à vérifier, au cas où vous n'obtiendriez pas les résultats escomptés.

- Vous avez reçu une bannière d'erreur

  Si une erreur est apparue lorsque vous avez publié le message d'entrée, corrigez-la d'abord. Les étapes suivantes peuvent vous aider à corriger cette erreur.

- Vous ne voyez pas le message d'entrée dans le MQTT client

  Chaque fois que vous publiez votre message d'entrée dans le `device/32/data` sujet, ce message doit apparaître dans le MQTT client, si vous vous êtes abonné au filtre de `device/+/data` sujet comme décrit dans la procédure.

  À savoir

  - Vérifiez le filtre de objets auquel vous vous êtes abonné

    Si vous vous êtes abonné au objet du message d'entrée comme décrit dans la procédure, vous devriez voir une copie du message d'entrée chaque fois que vous le publiez.

    Si le message ne s'affiche pas, vérifiez le nom de l'objet auquel vous vous êtes abonné et comparez-le au objet dans lequel vous avez publié. Les noms des objets distinguent les majuscules et minuscules et le objet auquel vous vous êtes abonné doit être identique au objet dans lequel vous avez publié la charge utile du message.

  - Vérifiez la fonction de publication des messages

    Dans le MQTT client, sous Abonnements, choisissez device/+/data, vérifiez le sujet du message de publication, puis choisissez Publier dans le sujet. La charge utile du message figurant dans la zone d'édition située sous le objet devrait apparaître dans la liste des messages.

- Vous ne recevez pas de SMS message

  Pour que votre règle fonctionne, elle doit disposer de la politique appropriée qui l'autorise à recevoir un message et à envoyer une SNS notification, et elle doit recevoir le message.

  À savoir

  - Vérifiez le Région AWS nom de votre MQTT client et la règle que vous avez créée

    La console dans laquelle vous exécutez le MQTT client doit se trouver dans la même AWS région que la règle que vous avez créée.

  - Vérifiez que la valeur de température dans la charge utile du message dépasse le seuil de test

    Si la valeur de température est inférieure ou égale à 30, telle que définie dans l'déclararionde requête de règle, la règle n'exécutera aucune de ses actions.

  - Vérifiez le objet du message d'entrée dans la déclaration de requête de règle

Pour que la règle fonctionne, elle doit recevoir un message dont le nom de rubrique correspond au filtre de rubrique figurant dans la FROM clause de l'instruction de requête de règle.

Vérifiez l'orthographe du filtre de rubrique dans l'instruction de requête de règle avec celle du sujet dans le MQTT client. Les noms de objets distinguent les majuscules et minuscules et le objet du message doit correspondre au filtre de objet indiqué dans l'déclararionde requête de règle.

- Vérifiez le contenu de la charge utile des messages d'entrée

  Pour que la règle fonctionne, elle doit trouver le champ de données dans la charge utile du message déclarée dans l'SELECTinstruction.

  Vérifiez l'orthographe du `temperature` champ dans l'instruction de requête de règle avec celle de la charge utile du message dans le MQTT client. Les noms de champs distinguent les majuscules et minuscules et le `temperature` champ de l'déclararionde requête de règle doit être identique au `temperature` champ de la charge du message.

  Assurez-vous que le JSON document contenu dans la charge utile du message est correctement formaté. S'il JSON contient des erreurs, telles qu'une virgule manquante, la règle ne pourra pas le lire.

- Consultez la SNS notification Amazon

  DansÉtape 1 : créer un SNS sujet Amazon qui envoie un message SMS texte, reportez-vous à l'étape 3 qui décrit comment tester la SNS notification Amazon et tester la notification pour vous assurer qu'elle fonctionne.

- La fonction Lambda.

  DansÉtape 1 : créer une AWS Lambda fonction qui envoie un message texte, reportez-vous à l'étape 5 qui décrit comment tester la fonction Lambda à l'aide de données de test et tester la fonction Lambda.

- Vérifiez le rôle utilisé par la règle

  L'action de règle doit être autorisée à recevoir le objet d'origine et à publier le nouveau objet.

  Les politiques qui autorisent la règle à recevoir les données des messages et à les republier sont spécifiques aux objets utilisés. Si vous modifiez le objet utilisé pour republier les données du message, vous devez mettre à jour le rôle de l'action de règle afin de mettre à jour sa politique afin qu'elle corresponde au objet actuel.

Si vous pensez que c'est le problème, modifiez l'action Republier la règle et créez un nouveau rôle. Les nouveaux rôles créés par l'action de règle reçoivent les autorisations nécessaires pour effectuer ces actions.

Étape 4 : examen des résultats et des étapes suivantes

Dans ce tutoriel :

- Vous avez créé une AWS IoT règle pour appeler une fonction Lambda qui a envoyé une SNS notification Amazon utilisant la charge utile de vos messages personnalisés.
- Vous avez utilisé une SQL requête et des fonctions simples dans une instruction de requête de règle pour créer une nouvelle charge utile de message pour votre fonction Lambda.
- Vous avez utilisé le MQTT client pour tester votre AWS IoT règle.

Étapes suivantes

Après avoir envoyé quelques SMS avec cette règle, essayez de l'utiliser pour voir comment la modification de certains aspects du didacticiel affecte le message et le moment où il est envoyé. Voici quelques idées pour vous aider à démarrer.

- Changez le *`device_id`* dans le sujet du message d'entrée et observez l'effet dans le contenu du message texte.
- Modifiez les champs sélectionnés dans l'déclararionde requête de règle, mettez à jour la fonction Lambda pour les utiliser dans un nouveau message et observez l'effet dans le contenu du message texte.
- Modifiez le test dans l'déclararionde requête de règle pour tester une température minimale au lieu d'une température maximale. Mettez à jour la fonction Lambda pour formater un nouveau message et n'oubliez pas de changer le nom de `max_temperature`.
- Pour en savoir plus sur la détection des erreurs susceptibles de se produire lors du développement et de l'utilisation de AWS IoT règles, consultezSurveillance AWS IoT.

Conservation de l'état de l'appareil lorsque celui-ci est hors ligne avec Device Shadows

Ces didacticiels vous montrent comment utiliser le service AWS IoT Device Shadow pour stocker et mettre à jour les informations d'état d'un appareil. Le document Shadow, qui est un document JSON, montre la modification de l'état de l'appareil en fonction des messages publiés par un appareil, une application locale ou un service. Dans ce didacticiel, le document Shadow montre le changement de couleur d'une ampoule. Ces didacticiels montrent également comment le shadow stocke ces informations même lorsque l'appareil est déconnecté d'Internet, et transmet les dernières informations d'état à l'appareil lorsqu'il revient en ligne et demande ces informations.

Nous vous recommandons d'essayer ces didacticiels dans l'ordre dans lequel ils sont présentés ici, en commençant par les AWS IoT ressources dont vous avez besoin pour créer et la configuration matérielle nécessaire, ce qui vous permet également d'apprendre les concepts progressivement. Ces didacticiels montrent comment configurer et connecter un appareil Raspberry Pi à utiliser avec AWS IoT. Si vous ne disposez pas du matériel requis, vous pouvez suivre ces didacticiels en les adaptant à l'appareil de votre choix ou en créant un appareil virtuel avec Amazon EC2.

Présentation des scénarios du didacticiel

Le scénario de ces didacticiels est une application ou un service local qui modifie la couleur d'une ampoule et publie ses données sur des sujets d'ombre réservés. Ces didacticiels sont similaires à la fonctionnalité Device Shadow décrite dans le didacticiel de démarrage interactif et sont implémentés sur un appareil Raspberry Pi. Les didacticiels de cette section se concentrent sur une seule ombre classique tout en montrant comment vous pouvez adapter des ombres nommées ou plusieurs appareils.

Les didacticiels suivants vous aideront à apprendre à utiliser le service AWS IoT Device Shadow.

- Tutoriel : Préparation de votre Raspberry Pi pour exécuter l'application fantôme

  Ce didacticiel explique comment configurer un appareil Raspberry Pi pour se connecter à AWS IoT. Vous allez également créer un document de AWS IoT stratégie et une ressource d'objet, télécharger les certificats, puis associer la politique à cette ressource d'objet. Ce didacticiel vous prendra environ 30 minutes.
- Tutoriel : Installation du SDK du périphérique et exécution de l'exemple d'application pour Device Shadows

Ce didacticiel explique comment installer les outils et logiciels requis, ainsi que le SDK AWS IoT Device pour Python, puis exécuter l'exemple d'application Shadow. Ce didacticiel s'appuie sur les concepts présentés dans le document Connectez un Raspberry Pi ou un autre appareil et dure 20 minutes.

- Tutoriel : Interaction avec Device Shadow à l'aide de l'exemple d'application et du client de test MQTT

  Ce didacticiel explique comment utiliser les `shadow.py` exemples d'application et de AWS IoT console pour observer l'interaction entre AWS IoT Device Shadows et les changements d'état de l'ampoule. Le didacticiel montre également comment envoyer des messages MQTT aux rubriques réservées du Device Shadow. Ce didacticiel vous prendra environ 45 minutes.

AWS IoT Présentation de Device Shadow

Un Device Shadow est une représentation virtuelle persistante d'un appareil géré par une ressource d'objets que vous créez dans le AWS IoT registre. Le document Shadow est un document JSON ou un document de JavaScript notation utilisé pour stocker et récupérer les informations d'état actuel d'un appareil. Vous pouvez utiliser l'ombre pour obtenir et définir l'état d'un appareil via des sujets MQTT ou HTTP REST APIs, que l'appareil soit connecté à Internet ou non.

Vous pouvez utiliser l'ombre pour obtenir et définir l'état d'un appareil via des sujets MQTT ou des API HTTP REST, que l'appareil soit ou non connecté à l'internet.

- Les applications spécifient les états souhaités des propriétés de l'appareil en mettant à jour `desired` l'objet.
- `reported` : Les appareils rapportent leur état actuel dans `reported` l'objet .
- `delta`: AWS IoT indique les différences entre l'état souhaité et l'état indiqué dans l'`delta`objet.

Voici un exemple de document de l'État fantôme.

```
{
  "state": {
    "desired": {
      "color": "green"
      },
    "reported": {
      "color": "blue"
```

```
      },
    "delta": {
      "color": "green"
      }
   }
}
```

Pour mettre à jour le document Shadow d'un appareil, vous pouvez utiliser les rubriques MQTT réservées, le Device Shadow REST APIs qui prend en charge le `GETUPDATE`, et les `DELETE` opérations avec HTTP, ainsi que la AWS IoT CLI.

Dans l'exemple précédent, supposons que vous souhaitiez changer la `desired` couleur en`yellow`. Pour ce faire, envoyez une demande à l'UpdateThingShadowAPI ou publiez un message dans la rubrique Mise à jour, `$aws/things/THING_NAME/shadow/update`.

```
{
  "state": {
    "desired": {
      "color": yellow
    }
  }
}
```

Les mises à jour concernent uniquement les champs spécifiés dans la demande. Après avoir correctement mis à jour le Device Shadow, le nouvel `desired` état est AWS IoT publié dans la `delta` rubrique`$aws/things/THING_NAME/shadow/delta`. Dans ce cas, le document Shadow ressemble à ceci :

```
{
  "state": {
    "desired": {
      "color": yellow
    },
    "reported": {
      "color": green
    },
    "delta": {
      "color": yellow
      }
  }
}
```

Le nouvel état est ensuite signalé au AWS IoT Device Shadow en utilisant le `Update` sujet `$aws/things/THING_NAME/shadow/update` avec le message JSON suivant :

```
{
  "state": {
    "reported": {
      "color": yellow
    }
  }
}
```

Si vous souhaitez obtenir les informations sur l'état actuel, envoyez une demande à l'GetThingShadowAPI ou publiez un message MQTT dans la rubrique Get,`$aws/things/THING_NAME/shadow/get`.

Pour plus d'informations, consultez Service d'ombre de périphérique dans le AWS IoT Service Device Shadow.

Pour plus d'informations sur l'utilisation de Device Shadows dans les appareils, les applications et les services, consultez Utilisation des shadows sur les appareils et Utilisation des shadows dans les applications et les services.

Pour plus d'informations sur l'interaction avec AWS IoT les ombres, voirInteraction avec les shadows.

Pour plus d'informations sur les sujets réservés MQTT et HTTP REST APIs, consultez Rubriques MQTT de Device Shadow etAPI REST Device Shadow.

Tutoriel : Préparation de votre Raspberry Pi pour exécuter l'application fantôme

Ce didacticiel explique comment installer et configurer un appareil Raspberry Pi et comment créer les AWS IoT ressources dont un appareil a besoin pour se connecter et échanger des messages MQTT.

> **Note**
>
> Si vous avez l'intention de the section called “Créez un appareil virtuel avec Amazon EC2”, vous pouvez ignorer cette page et passer à the section called “Configurer votre appareil”. Vous créerez ces ressources lorsque vous créerez votre objet virtuel. Si vous souhaitez utiliser un autre appareil à la place du Raspberry Pi, vous pouvez essayer de suivre ces tutoriels en les adaptant à l'appareil de votre choix.

Dans ce didacticiel, vous allez découvrir comment :

- Installez un appareil Raspberry Pi et configurez-le pour une utilisation avec AWS IoT.
- Créez un document AWS IoT de politique qui autorise votre appareil à interagir avec les AWS IoT services.
- Créez une ressource d'objet dans AWS IoT les certificats de périphérique X.509, puis joignez le document de politique.

  Le truc, c'est la représentation virtuelle de votre appareil dans le AWS IoT registre. Le certificat authentifie votre appareil auprès de AWS IoT Core, et le document de politique autorise l'interaction avec votre appareil. AWS IoT

Comment exécuter ce didacticiel

Pour exécuter l'`shadow.py`exemple d'application pour Device Shadows, vous aurez besoin d'un appareil Raspberry Pi connecté à AWS IoT. Nous vous recommandons de suivre ce didacticiel dans l'ordre dans lequel il est présenté ici, en commençant par configurer le Raspberry Pi et ses accessoires, puis en créant une politique et en l'attachant à une ressource d'objet que vous créez. Vous pouvez ensuite suivre ce didacticiel en utilisant l'interface utilisateur graphique (GUI) prise en charge par le Raspberry Pi pour ouvrir la AWS IoT console sur le navigateur Web de l'appareil, ce qui facilite également le téléchargement des certificats directement sur votre Raspberry Pi pour vous y connecter AWS IoT.

Avant de commencer ce didacticiel, assurez-vous de disposer des éléments suivants :

- Un Compte AWS. Si vous n'en avez pas, effectuez les étapes décrites dans Configurez Compte AWS avant de continuer. Vous aurez besoin de votre AWS IoT console Compte AWS et de votre console pour terminer ce didacticiel.
- Le Raspberry Pi et ses accessoires nécessaires. Vous aurez besoin de :
  - Un Raspberry Pi 3 modèle B ou un modèle plus récent. Ce tutoriel peut fonctionner sur des versions antérieures du Raspberry Pi, mais nous ne l'avons pas testé.
  - Système d'exploitation Raspberry Pi (32 bits) ou version ultérieure. Nous vous recommandons d'utiliser la dernière version du système d'exploitation Raspberry Pi. Les versions antérieures du système d'exploitation peuvent fonctionner, mais nous ne les avons pas testées.
  - Une connexion Ethernet ou Wi-Fi.
  - Clavier, souris, écran, câbles et blocs d'alimentation.

Ce didacticiel vous prendra environ 30 minutes.

Étape 1 : Installation et configuration de l'appareil Raspberry Pi

Dans cette section, nous allons configurer un appareil Raspberry Pi à utiliser avec AWS IoT.

> **Important**
>
> L'adaptation de ces instructions à d'autres appareils et systèmes d'exploitation peut s'avérer difficile. Vous devez avoir une connaissance suffisante de votre appareil pour être en mesure d'interpréter ces instructions et de les appliquer. Si vous rencontrez des difficultés, vous pouvez essayer l'une des autres options de l'appareil comme alternative, telle que Créez un appareil virtuel avec Amazon EC2 ouUtilisez votre PC ou Mac Windows ou Linux comme AWS IoT appareil.

Vous devez configurer votre Raspberry Pi de manière à ce qu'il puisse démarrer le système d'exploitation (OS), se connecter à Internet et vous permettre d'interagir avec lui via une interface en ligne de commande. Vous pouvez également utiliser l'interface utilisateur graphique (GUI) compatible avec le Raspberry Pi pour ouvrir la AWS IoT console et exécuter le reste de ce didacticiel.

Pour configurer le Raspberry Pi

1. Insérez la carte SD dans le port MicroSD dans le Raspberry Pi. Certaines cartes SD sont préchargées avec un gestionnaire d'installation qui vous invite dans un menu à installer le système d'exploitation après le démarrage de la carte. Vous pouvez également utiliser l'imageur Raspberry Pi pour installer le système d'exploitation sur votre carte.
2. Connectez un téléviseur ou un moniteur HDMI au câble HDMI qui se connecte au port HDMI du Raspberry Pi.
3. Connectez le clavier et la souris aux ports USB du Raspberry Pi, puis branchez l'adaptateur secteur pour démarrer la carte.

Après le démarrage du Raspberry Pi, si la carte SD a été préchargée avec le gestionnaire d'installation, un menu apparaît pour installer le système d'exploitation. Si vous rencontrez des difficultés lors de l'installation du système d'exploitation, vous pouvez essayez de suivre la procédure ci-dessous. Pour plus d'informations sur la configuration de votre Raspberry Pi, consultez le Guide de démarrage Raspberry Pi.

Si vous ne parvenez pas à configurer le Raspberry Pi :

- Vérifiez si vous avez inséré la carte SD avant de démarrer la carte. Si vous branchez la carte SD après avoir démarré la carte, le menu d'installation risque de ne pas apparaître.
- Assurez-vous que le téléviseur ou le moniteur est allumé et que l'entrée correcte est sélectionnée.
- Assurez-vous que vous utilisez un logiciel compatible avec le Raspberry Pi.

Après avoir installé et configuré le système d'exploitation Raspberry Pi, ouvrez le navigateur Web du Raspberry Pi et accédez à la AWS IoT Core console pour poursuivre les étapes suivantes de ce didacticiel.

Si vous pouvez ouvrir la AWS IoT Core console, votre Raspberry Pi est prêt et vous pouvez continuerthe section called “Approvisionnement de votre appareil dans AWS IoT ”.

Si vous rencontrez des problèmes ou avez besoin d'une aide supplémentaire, consultez Obtenir de l'aide pour votre Raspberry Pi.

Tutoriel : provisionnement de votre appareil dans AWS IoT

Cette section crée les AWS IoT Core ressources que votre didacticiel utilisera.

Étapes à suivre pour approvisionner votre appareil AWS IoT

- Étape 1 : créer une AWS IoT politique pour le Device Shadow
- Étape 2 : créer une ressource d'objet et associer la politique à l'objet
- Étape 3 : examen des résultats et des étapes suivantes

Étape 1 : créer une AWS IoT politique pour le Device Shadow

Les certificats X.509 authentifient votre appareil avec. AWS IoT Core AWS IoT des politiques sont associées au certificat qui permettent à l'appareil d'effectuer AWS IoT des opérations, telles que l'abonnement ou la publication de sujets réservés au MQTT utilisés par le service Device Shadow. Votre appareil présente son certificat lorsqu'il se connecte et envoie des messages à AWS IoT Core.

Dans cette procédure, vous allez créer une politique qui permet à votre appareil d'effectuer les AWS IoT opérations nécessaires pour exécuter le programme d'exemple. Nous vous recommandons de créer une politique qui n'accorde que les autorisations nécessaires à l'exécution de la tâche. Vous créez d'abord la AWS IoT politique, puis vous l'associez au certificat de terminal que vous créerez ultérieurement.

Pour créer une AWS IoT politique

1. Dans le panneau de navigation, choisissez Sécurité, puis Stratégies. Si des politiques existent déjà dans votre compte, choisissez Créer, sinon, sur la page Vous n'avez pas encore de politique, choisissez Créer une politique.
2. Sur la page Create a policy (Créer une politique) :

   a. Dans le champ Nom entrez un nom pour la politique (par exemple, **`My_Device_Shadow_policy`**). N'utilisez pas d'informations personnelles identifiables dans vos noms de stratégie.

   b. Dans le document de politique, vous décrivez les actions de connexion, d'abonnement, de réception et de publication qui donnent à l'appareil l'autorisation de publier et de s'abonner aux sujets réservés au MQTT.

      Copiez l'exemple de politique suivant et collez-le dans votre document de politique. Remplacez-le `thingname` par le nom de l'objet que vous allez créer (par exemple,`My_light_bulb`), `region` par la AWS IoT région dans laquelle vous utilisez les services et `account` par votre Compte AWS numéro. Pour plus d'informations sur AWS IoT les politiques, consultezAWS IoT Core politiques.

      JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/
get",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/
update"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/
get/accepted",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/
get/rejected",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/
update/accepted",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/
update/rejected",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/
update/delta"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/
shadow/get/accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/
shadow/get/rejected",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/
shadow/update/accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/
shadow/update/rejected",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/
shadow/update/delta"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:region:account:client/test-*"
    }
  ]
}
```

Étape 2 : créer une ressource d'objet et associer la politique à l'objet

Les appareils connectés à AWS IoT peuvent être représentés par des ressources d'objets dans le AWS IoT registre. Une ressource d'objet représente un périphérique ou une entité logique spécifique, comme l'ampoule dans ce didacticiel.

Pour savoir comment créer un objet dans AWS IoT, suivez les étapes décrites dansCréez un objet. Voici quelques points essentiels à noter lorsque vous suivez les étapes de ce didacticiel :

1. Choisissez Créer un objet unique, puis dans le champ Nom, entrez un nom pour l'objet identique à celui `thingname` (par exemple`My_light_bulb`) que vous avez spécifié lors de la création de la politique précédemment.

   Il n'est pas possible de modifier le nom d'une chose après sa création. Si vous lui avez donné un autre nom que `thingname`, créez un nouvel objet avec le nom as `thingname` et supprimez l'ancien.

   > **Note**
   >
   > N'utilisez pas d'informations personnellement identifiables dans votre nom de domaine. Le nom de l'objet peut apparaître dans les communications et les rapports non chiffrés.

2. Nous vous recommandons de télécharger chacun des fichiers de certificat figurant sur le certificat créé ! page vers un endroit où vous pouvez facilement les trouver. Vous devez installer ces fichiers pour exécuter l'exemple d'application.

   Nous vous recommandons de télécharger les fichiers dans un `certs` sous-répertoire de votre home répertoire sur le Raspberry Pi et de nommer chacun d'eux avec un nom plus simple, comme suggéré dans le tableau suivant.

   Noms des fichiers de certificat

Fichier	Chemin d'accès du fichier
Certificat racine de l'autorité de certification	`~/certs/Amazon-root-CA-1.pem`
Certificat de l'appareil	`~/certs/device.pem.crt`
Clé privée	`~/certs/private.pem.key`

3. Après avoir activé le certificat pour autoriser les connexions AWS IoT, choisissez Attacher une politique et assurez-vous d'associer la politique que vous avez créée précédemment (par exemple,**`My_Device_Shadow_policy`**) à l'objet.

   Une fois que vous avez créé un objet, vous pouvez voir sa ressource s'afficher dans la liste des objets de la AWS IoT console.

Étape 3 : examen des résultats et des étapes suivantes

Dans ce didacticiel, vous allez découvrir comment :

- Installation et configuration de l'appareil Raspberry Pi.
- Créez un document AWS IoT de politique qui autorise votre appareil à interagir avec les AWS IoT services.
- Créez une ressource d'objet et le certificat de périphérique X.509 associé, puis joignez-y le document de politique.

Étapes suivantes

Vous pouvez désormais installer le SDK du AWS IoT périphérique pour Python, exécuter l'`shadow.py`exemple d'application et utiliser Device Shadows pour contrôler l'état. Pour de plus amples informations sur sa configuration, veuillez consulter Tutoriel : Installation du SDK du périphérique et exécution de l'exemple d'application pour Device Shadows.

Tutoriel : Installation du SDK du périphérique et exécution de l'exemple d'application pour Device Shadows

Cette section explique comment installer le logiciel requis et le SDK AWS IoT Device pour Python et exécuter l'`shadow.py`exemple d'application pour modifier le document Shadow et contrôler l'état de l'ombre.

Dans ce didacticiel, vous allez découvrir comment :

- Utilisez le logiciel installé et le SDK AWS IoT Device pour Python pour exécuter l'exemple d'application.
- Découvrez comment la saisie d'une valeur à l'aide de l'exemple d'application permet de publier la valeur souhaitée dans la AWS IoT console.

- Passez en revue l'`shadow.py`exemple d'application et découvrez comment elle utilise le protocole MQTT pour mettre à jour l'état de l'ombre.

Avant de lancer ce didacticiel :

Vous devez avoir configuré votre Compte AWS appareil Raspberry Pi, et créé un AWS IoT élément et une politique qui donnent à l'appareil l'autorisation de publier et de s'abonner aux sujets réservés MQTT du service Device Shadow. Pour de plus amples informations, veuillez consulter Tutoriel : Préparation de votre Raspberry Pi pour exécuter l'application fantôme.

Vous devez également avoir installé Git, Python et le AWS IoT Device SDK pour Python. Ce didacticiel s'appuie sur les concepts présentés dans le didacticielConnectez un Raspberry Pi ou un autre appareil. Si vous n'avez pas essayé ce didacticiel, nous vous recommandons de suivre les étapes décrites dans ce didacticiel pour installer les fichiers de certificat et le SDK du périphérique, puis de revenir à ce didacticiel pour exécuter l'`shadow.py`exemple d'application.

Dans ce tutoriel, vous allez :

- Étape 1 : Exécutez l'exemple d'application shadow.py
- Étape 2 : passez en revue l'exemple d'application shadow.py Device SDK
- Étape 3 : résolution des problèmes liés à l'shadow.pyexemple d'application
- Étape 4 : examen des résultats et des étapes suivantes

Ce didacticiel vous prendra environ 20 minutes.

Étape 1 : Exécutez l'exemple d'application shadow.py

Avant d'exécuter l'`shadow.py`exemple d'application, vous aurez besoin des informations suivantes, en plus des noms et de l'emplacement des fichiers de certificats que vous avez installés.

Valeurs des paramètres de l'application

Paramètre	Où trouver la valeur
`your-iot-thing-name`	Nom de l' AWS IoT objet que vous avez créé plus tôt dansthe section called “Étape 2 : créer une ressource d'objet et associer la politique à l'objet”.

Paramètre	Où trouver la valeur
	Pour trouver cette valeur, dans la AWS IoT console, choisissez Gérer, puis Objets.
your-iot-endpoint	La *your-iot-endpoint* valeur a le format suivant :*endpoint_id* - ats.iot. *region*.amazonaws.com , par exemple,a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com . Utilisez pour trouver cette valeur. 1. Dans la AWS IoT console, choisissez Gérer, puis Tâches. 2. Choisissez l'objet IoT que vous avez créé pour votre appareil, My_light_bulb, que vous avez utilisé précédemment, puis choisisse z Interact. Sur la page des détails de l'objet votre point de terminaison s'affiche dans la section HTTPS

Installez et exécutez l'exemple d'application

1. Accédez au répertoire de l'exemple d'application.

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

2. Dans la fenêtre de ligne de commande, remplacez *your-iot-endpoint* et *your-iot-thing-name* comme indiqué et exécutez cette commande.

```
python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
thing_name your-iot-thing-name
```

3. Notez que l'exemple d'application :

   1. Se connecte au service AWS IoT associé à votre compte.
   2. S'abonne aux `Delta` événements et `Update` et `Get` aux réponses.

3. Vous invite à saisir la valeur souhaitée dans le terminal.
4. Affiche une sortie similaire à celle-ci :

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to Delta events...
Subscribing to Update responses...
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow contains reported value 'off'.
Enter desired value:
```

> **Note**
>
> Si vous ne parvenez pas à exécuter l'`shadow.py`exemple d'application, consultezthe section called “Étape 3 : résolution des problèmes liés à l'`shadow.py`exemple d'application”. Pour obtenir des informations supplémentaires susceptibles de vous aider à corriger le problème, ajoutez le `--verbosity debug` paramètre sur la ligne de commande afin que l'exemple d'application affiche des messages détaillés sur ce qu'elle fait.

Entrez des valeurs et observez les mises à jour dans le document Shadow

Vous pouvez entrer des valeurs dans le terminal pour spécifier la `desired` valeur, qui met également à jour la `reported` valeur. Supposons que vous saisissiez la couleur `yellow` dans le terminal. La `reported` valeur est également mise à jour en fonction de la couleur`yellow`. Les messages affichés dans le terminal sont les suivants :

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

Lorsque vous publiez cette demande de mise à jour, elle AWS IoT crée une ombre classique par défaut pour la ressource objet. Vous pouvez observer la demande de mise à jour que vous avez publiée sur `desired` les valeurs `reported` et de la AWS IoT console en consultant le document Shadow correspondant à la ressource d'objet que vous avez créée (par exemple,`My_light_bulb`). Pour voir la mise à jour dans le document Shadow :

1. Dans la AWS IoT console, choisissez Gérer, puis Objets.
2. Dans la liste des éléments affichés, sélectionnez l'objet que vous avez créé, choisissez Ombre, puis Ombre classic.

Le document Shadow doit ressembler à ce qui suit, avec les `desired` valeurs `reported` et définies en fonction de la couleur`yellow`. Vous pouvez voir ces valeurs dans la section Shadow state du document.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "yellow"
},
"reported": {
  "welcome": "aws-iot",
  "color": "yellow"
}
}
```

Vous pouvez également voir une section de métadonnées qui contient les informations d'horodatage et le numéro de version de la demande.

Vous pouvez utiliser la version du document d'état pour vous assurer que vous mettez à jour la version la plus récente du document d'ombre d'un appareil. Si vous envoyez une autre demande de mise à jour, le numéro de version augmente de 1. Lorsque vous fournissez une version avec une demande de mise à jour, le service rejette la demande avec un code de réponse de conflit HTTP 409 si la version actuelle du document d'état ne correspond pas à la version fournie.

```
{
"metadata": {
  "desired": {
    "welcome": {
      "timestamp": 1620156892
```

```
    },
    "color": {
      "timestamp": 1620156893
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  }
},
"version": 10
}
```

Pour en savoir plus sur le document Shadow et observer les modifications apportées aux informations d'état, passez au didacticiel suivant, Tutoriel : Interaction avec Device Shadow à l'aide de l'exemple d'application et du client de test MQTT comme décrit dans la Étape 4 : examen des résultats et des étapes suivantes section de ce didacticiel. Vous pouvez éventuellement en apprendre davantage sur l'`shadow.py`exemple de code et sur la façon dont il utilise le protocole MQTT dans la section suivante.

Étape 2 : passez en revue l'exemple d'application shadow.py Device SDK

Cette section passe en revue l'`shadow.py`exemple d'application du AWS IoT Device SDK v2 pour Python utilisé dans ce didacticiel. Ici, nous allons voir comment il se connecte à l'aide AWS IoT Core des protocoles MQTT et MQTT over WSS. La bibliothèque AWS Common Runtime (AWS-CRT) fournit le support des protocoles de communication de bas niveau et est incluse dans le AWS IoT Device SDK v2 pour Python.

Bien que ce didacticiel utilise MQTT et MQTT sur WSS, il AWS IoT prend en charge les appareils qui publient des requêtes HTTPS. Pour un exemple de programme Python qui envoie un message HTTP depuis un appareil, consultez l'exemple de code HTTPS utilisant la `requests` bibliothèque Python.

Pour savoir comment prendre une décision éclairée quant au protocole à utiliser pour les communications de votre appareil, consultez le Choix d'un protocole d'application pour la communication de votre appareil.

MQTT

Les `shadow.py` exemples d'appels `mtls_from_path` (présentés ici) dans le `mqtt_connection_builder`pour établir une connexion à AWS IoT Core l'aide du protocole MQTT. `mtls_from_path` utilise les certificats X.509 et le protocole TLS v1.2 pour authentifier le périphérique. La bibliothèque AWS-CRT gère les détails de niveau inférieur de cette connexion.

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
  endpoint=args.endpoint,
  cert_filepath=args.cert,
  pri_key_filepath=args.key,
  ca_filepath=args.ca_file,
  client_bootstrap=client_bootstrap,
  on_connection_interrupted=on_connection_interrupted,
  on_connection_resumed=on_connection_resumed,
  client_id=args.client_id,
  clean_session=False,
  keep_alive_secs=6
)
```

- `endpoint`est votre AWS IoT point de terminaison que vous avez transmis depuis la ligne de commande et `client_id` est l'identifiant qui identifie de manière unique cet appareil dans le Région AWS.
- `cert_filepath`, `pri_key_filepath`, et `ca_filepath` sont les chemins d'accès aux fichiers de certificat et de clé privée de l'appareil, ainsi qu'au fichier CA racine.
- `client_bootstrap` est l'objet d'exécution courant qui gère les activités de communication par socket et qui est instancié avant l'appel à `mqtt_connection_builder.mtls_from_path`.
- `on_connection_interrupted` et `on_connection_resumed` sont des fonctions de rappel à appeler lorsque la connexion de l'appareil est interrompue et reprise.
- `clean_session` est de savoir s'il faut démarrer une nouvelle session persistante ou, s'il y en a une, se reconnecter à une session existante. `keep_alive_secs` est la valeur de maintien en vie, en secondes, à envoyer à la `CONNECT` demande. Un ping sera automatiquement envoyé à cet intervalle. Le serveur suppose que la connexion est perdue s'il ne reçoit pas de ping après 1,5 fois cette valeur.

L'`shadow.py` exemple fait également appel `websockets_with_default_aws_signing` au `mqtt_connection_builder` pour établir une connexion en AWS IoT Core utilisant le protocole MQTT via WSS. MQTT over WSS utilise également les mêmes paramètres que MQTT et prend les paramètres supplémentaires suivants :

- `region`est la région de AWS signature utilisée par l'authentification Signature V4 et `credentials_provider` les AWS informations d'identification fournies pour l'authentification. La région est transmise depuis la ligne de commande et l'`credentials_provider`objet est instancié juste avant l'appel à. `mqtt_connection_builder.websockets_with_default_aws_signing`
- `websocket_proxy_options` est l'option du proxy HTTP, si vous utilisez un hôte proxy. Dans l'`shadow.py`exemple d'application, cette valeur est instanciée juste avant l'appel à `mqtt_connection_builder.websockets_with_default_aws_signing`.

Abonnez-vous aux sujets et événements du Shadow

L'`shadow.py`échantillon tente d'établir une connexion et attend d'être complètement connecté. S'il n'est pas connecté, les commandes sont mises en file d'attente. Une fois connecté, l'échantillon s'abonne aux événements delta, met à jour et reçoit des messages, et publie des messages avec un niveau de qualité de service (QoS) de 1 (`mqtt.QoS.AT_LEAST_ONCE`).

Lorsqu'un appareil s'abonne à un message avec QoS de niveau 1, le courtier de messages enregistre les messages auxquels le périphérique est abonné jusqu'à ce qu'ils puissent être envoyés au terminal. Le courtier de messages renvoie les messages jusqu'à ce qu'il reçoive une PUBACK réponse de l'appareil.

Pour plus d'informations sur les procédures stockées, consultez Passez en revue le MQTT protocole et MQTT.

Pour plus d'informations sur la façon dont MQTT, MQTT over WSS, les sessions persistantes et les niveaux de QoS utilisés dans ce didacticiel, consultez Consultez l'SDKexemple d'application pubsub.py pour appareil.

Étape 3 : résolution des problèmes liés à l'**`shadow.py`**exemple d'application

Lorsque vous exécutez l'`shadow.py`exemple d'application, certains messages devraient s'afficher dans le terminal et vous demander de saisir une `desired` valeur. Si le programme génère une erreur, alors pour corriger l'erreur, vous pouvez commencer par vérifier si vous avez exécuté la bonne commande pour votre système.

Dans certains cas, le message d'erreur peut indiquer des problèmes de connexion et ressembler à : `Host name was invalid for dns resolution` ou`Connection was closed unexpectedly`. Dans de tels cas, voici certaines choses que vous pouvez vérifier :

- Vérifiez l'adresse du point de terminaison dans la commande

  Vérifiez l'`endpoint`argument de la commande que vous avez saisie pour exécuter l'exemple d'application (par exemple,`a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`) et vérifiez cette valeur dans la AWS IoT console.

  Pour vérifier si vous avez utilisé la bonne valeur :

  1. Dans la AWS IoT console, choisissez Gérer, puis Tâches.
  2. Choisissez l'élément que vous avez créé pour votre exemple d'application (par exemple, My_light_bulb), puis choisissez Interact.

  Sur la page des détails de l'objet votre point de terminaison s'affiche dans la section HTTPS Vous devriez également voir un message indiquant : `This thing already appears to be connected.`

- Vérifier l'activation du certificat

  Les certificats authentifient votre appareil avec AWS IoT Core.

  Pour vérifier si votre certificat est actif, procédez comme suit :

  1. Dans la AWS IoT console, choisissez Gérer, puis Tâches.
  2. Choisissez l'élément que vous avez créé pour votre exemple d'application (par exemple, My_light_bulb), puis sélectionnez Security.
  3. Sélectionnez le certificat, puis, sur la page de détails du certificat, choisissez Sélectionner le certificat puis, sur la page de détails du certificat, choisissez Actions.

  Si l'option Activer n'est pas disponible dans la liste déroulante et que vous pouvez uniquement sélectionner Désactiver, votre certificat est actif. Si ce n'est pas le cas, choisissez Activer et réexécutez l'exemple de programme.

  Si le programme ne s'exécute toujours pas, vérifiez les noms des fichiers de certificats dans le `certs` dossier.

- Vérifiez la politique attachée à la ressource Thing

  Alors que les certificats authentifient votre appareil, AWS IoT les politiques permettent à l'appareil d'effectuer AWS IoT des opérations, telles que l'abonnement ou la publication sur des sujets réservés au MQTT.

  Pour vérifier si la bonne politique est jointe, procédez comme suit :

1. Recherchez le certificat comme décrit précédemment, puis choisissez Policies.
2. Choisissez la politique affichée et vérifiez si elle décrit les`connect`, `subscribe`, `receive`, et `publish` actions qui autorisent l'appareil à publier et à s'abonner aux sujets réservés au MQTT.

   Consultez Étape 1 : créer une AWS IoT politique pour le Device Shadow pour obtenir un exemple de stratégie.

Si vous voyez des messages d'erreur indiquant un problème de connexion AWS IoT, cela peut être dû aux autorisations que vous utilisez pour la politique. Dans ce cas, nous vous recommandons de commencer par une politique qui fournit un accès complet aux AWS IoT ressources, puis de réexécuter l'exemple de programme. Vous pouvez soit modifier la stratégie actuelle, soit choisir la stratégie actuelle, choisir Détacher, puis créer une autre politique fournissant un accès complet et l'associer à votre ressource d'objet. Vous pouvez ensuite restreindre la politique aux seules actions et politiques dont vous avez besoin pour exécuter le programme.

JSON

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

- Vérifiez l'installation du SDK de votre appareil

  Si le programme ne s'exécute toujours pas, vous pouvez réinstaller le SDK du périphérique pour vous assurer que l'installation du SDK est complète et correcte.

Étape 4 : examen des résultats et des étapes suivantes

Dans ce didacticiel, vous allez découvrir comment :

- Installez le logiciel, les outils et le SDK AWS IoT Device pour Python requis.

- Découvrez comment l'exemple d'application, `shadow.py`, le protocole MQTT pour récupérer et mettre à jour l'état actuel de l'ombre.
- Exécutez l'exemple d'application pour Device Shadows et observez la mise à jour du document Shadow dans la AWS IoT console. Vous avez également appris à résoudre les problèmes et à corriger les erreurs lors de l'exécution du programme.

Étapes suivantes

Vous pouvez désormais exécuter l'`shadow.py`exemple d'application et utiliser Device Shadows pour contrôler l'état. Vous pouvez observer les mises à jour apportées au document Shadow dans la AWS IoT console et observer les événements delta auxquels l'exemple d'application répond. À l'aide du client de test MQTT, vous pouvez vous abonner aux sujets secondaires réservés et observer les messages reçus par les sujets lors de l'exécution de l'exemple de programme. Pour de plus amples informations sur sa configuration veuillez consulter Tutoriel : Interaction avec Device Shadow à l'aide de l'exemple d'application et du client de test MQTT.

Tutoriel : Interaction avec Device Shadow à l'aide de l'exemple d'application et du client de test MQTT

Pour interagir avec l'`shadow.py`exemple d'application, entrez une valeur dans le terminal pour cette `desired` valeur. Par exemple, vous pouvez spécifier des couleurs qui ressemblent aux feux de signalisation, AWS IoT répondre à la demande et mettre à jour les valeurs signalées.

Dans ce didacticiel, vous allez découvrir comment :

- Utilisez l'`shadow.py`exemple d'application pour spécifier les états souhaités et mettre à jour l'état actuel de l'ombre.
- Modifiez le document Shadow pour observer les événements delta et la façon dont l'`shadow.py`exemple d'application y répond.
- Utilisez le client de test MQTT pour vous abonner à des sujets fictifs et observer les mises à jour lorsque vous exécutez l'exemple de programme.

Avant d'exécuter ce didacticiel, vous devez disposer des éléments suivants :

Configurez votre Compte AWS appareil Raspberry Pi, configurez votre appareil et créez AWS IoT quelque chose et une politique. Vous devez également avoir installé le logiciel requis, le SDK du périphérique, les fichiers de certificat et exécuté l'exemple de programme dans le terminal. Pour

de plus amples informations, veuillez consulter Tutoriel : Préparation de votre Raspberry Pi pour exécuter l'application fantôme et les didacticiels et Étape 1 : Exécutez l'exemple d'application shadow.py. Vous devez suivre ces didacticiels si ce n'est pas déjà fait.

Dans ce tutoriel, vous allez :

- Étape 1 : Mettre à jour les valeurs souhaitées et signalées à l'aide d'shadow.pyun exemple d'application
- Étape 2 : Afficher les messages de l'shadow.pyexemple d'application dans le client de test MQTT
- Étape 3 : Résolution des erreurs liées aux interactions entre Device Shadow et Device Shadow
- Étape 4 : examen des résultats et des étapes suivantes

Ce didacticiel vous prendra environ 45 minutes.

Étape 1 : Mettre à jour les valeurs souhaitées et signalées à l'aide d'**`shadow.py`**un exemple d'application

Dans le didacticiel précédentÉtape 1 : Exécutez l'exemple d'application shadow.py, vous avez appris à observer un message publié dans le document Shadow dans la AWS IoT console lorsque vous entrez une valeur souhaitée, comme décrit dans la sectionTutoriel : Installation du SDK du périphérique et exécution de l'exemple d'application pour Device Shadows.

Dans l'exemple précédent, nous avons défini la couleur souhaitée sur `yellow`. Après avoir saisi chaque valeur, le terminal vous invite à en saisir une autre`desired` valeur. Si vous entrez à nouveau la même valeur (`yellow`), l'application le reconnaît et vous invite à saisir une nouvelle `desired` valeur.

```
Enter desired value:
yellow
Local value is already 'yellow'.
Enter desired value:
```

Maintenant, disons que vous entrez la couleur`green`. AWS IoT répond à la demande et met à jour la `reported` valeur en`green`. C'est ainsi que la mise à jour se produit lorsque l'`desired`état est différent de l'`reported`état, provoquant un delta.

Comment l'**`shadow.py`**exemple d'application simule les interactions avec Device Shadow :

1. Entrez une `desired` valeur (disons`yellow`) dans le terminal pour publier l'état souhaité.

2. Comme l'`desired`état est différent de l'`reported`état (disons la couleur`green`), un delta se produit et l'application abonnée au delta reçoit ce message.
3. L'application répond au message et met à jour son état en fonction de la `desired` valeur, `yellow`.
4. L'application publie ensuite un message de mise à jour avec la nouvelle valeur signalée de l'état de l'appareil, `yellow`.

Vous trouverez ci-dessous les messages affichés dans le terminal indiquant comment la demande de mise à jour est publiée.

```
Enter desired value:
green
Changed local shadow value to 'green'.
Updating reported shadow value to 'green'...
Update request published.
Finished updating reported shadow value to 'green'.
```

Dans la AWS IoT console, le document Shadow reflète la valeur mise à jour `green` pour les `desired` champs `reported` et, et le numéro de version est incrémenté de 1. Par exemple, si le numéro de version précédent était affiché sous la forme 10, le numéro de version actuel sera affiché sous la forme 11.

> **Note**
>
> La suppression d'une ombre ne rétablit pas le numéro de version à 0. Vous verrez que la version fantôme est incrémentée de 1 lorsque vous publiez une demande de mise à jour ou que vous créez une autre version fantôme portant le même nom.

Modifiez le document Shadow pour observer les événements du delta

L'`shadow.py`exemple d'application est également abonnée aux `delta` événements et répond lorsque la `desired` valeur est modifiée. Par exemple, vous pouvez remplacer la `desired` valeur par la couleur`red`. Pour ce faire, dans la AWS IoT console, modifiez le document Shadow en cliquant sur Modifier, puis définissez la `desired` valeur sur `red` dans le JSON, tout en conservant la `reported` valeur sur`green`. Avant d'enregistrer les modifications, laissez le terminal du Raspberry Pi ouvert, car des messages s'afficheront sur le terminal lorsque le changement se produira.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

Après avoir enregistré la nouvelle valeur, l'`shadow.py`exemple d'application répond à cette modification et affiche des messages dans le terminal indiquant le delta. Vous devriez alors voir les messages suivants apparaître sous l'invite de saisie de la `desired` valeur.

```
Enter desired value:
Received shadow delta event.
Delta reports that desired value is 'red'. Changing local value...
Changed local shadow value to 'red'.
Updating reported shadow value to 'red'...
Finished updating reported shadow value to 'red'.
Enter desired value:
Update request published.
Finished updating reported shadow value to 'red'.
```

Étape 2 : Afficher les messages de l'**`shadow.py`**exemple d'application dans le client de test MQTT

Vous pouvez utiliser le client de test MQTT dans la AWS IoT console pour surveiller les messages MQTT transmis à votre Compte AWS. En vous abonnant à des sujets MQTT réservés utilisés par le service Device Shadow, vous pouvez observer les messages reçus par les sujets lors de l'exécution de l'exemple d'application.

Si vous n'avez pas encore utilisé le client de test MQTT, vous pouvez le consulter Afficher les messages MQTT avec le client AWS IoT MQTT. Cela vous permet d'apprendre à utiliser le client de test MQTT dans la AWS IoT console pour afficher les messages MQTT lorsqu'ils transitent par l'agent de messages.

1. Ouvrez le client de test MQTT

   Ouvrez le client de test MQTT dans la AWS IoT console dans une nouvelle fenêtre afin de pouvoir observer les messages reçus par les sujets MQTT sans perdre la configuration de votre

client de test MQTT. Le client de test MQTT ne conserve aucun abonnement ou journal de messages si vous le quittez pour accéder à une autre page de la console. Dans cette section du didacticiel, vous pouvez ouvrir le document Shadow correspondant à votre appareil et AWS IoT le client de test MQTT dans des fenêtres séparées afin d'observer plus facilement l'interaction avec Device Shadows.

2. Abonnez-vous aux Shadow topics réservés au MQTT

   Vous pouvez utiliser le client de test MQTT pour saisir les noms des sujets réservés au MQTT de Device Shadow et vous y abonner pour recevoir des mises à jour lors de l'exécution de l'`shadow.py`exemple d'application. Pour s'abonner à la rubrique MQTT

   a. Dans le client de test MQTT de la AWS IoT console, choisissez S'abonner à un sujet.

   b. Dans la section Filtre par sujet, entrez : ***`thingname`***$aws/things/ /shadow/update/ #. `thingname`Voici le nom de la ressource d'objets que vous avez créée précédemment (par exemple,`My_light_bulb`).

   c. Conservez les valeurs par défaut pour les paramètres de configuration supplémentaires, puis choisissez Subscribe.

   En utilisant le caractère générique # dans l'abonnement aux sujets, vous pouvez vous abonner à plusieurs sujets MQTT en même temps et observer tous les messages échangés entre l'appareil et son Shadow dans une seule fenêtre. Pour plus d'informations sur les caractères génériques et leur utilisation de, consultezRubriques MQTT.

3. Exécutez **`shadow.py`** un exemple de programme et observez les messages

   Dans la fenêtre de ligne de commande du Raspberry Pi, si vous avez déconnecté le programme, réexécutez l'exemple d'application et regardez les messages du client de test MQTT dans la AWS IoT console.

   a. Exécutez la commande suivante pour redémarrer le programme d'exemple. Remplacez *`your-iot-thing-name`* et *`your-iot-endpoint`* par les noms de l' AWS IoT objet que vous avez créé précédemment (par exemple,`My_light_bulb`) et du point de terminaison pour interagir avec l'appareil.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
   device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
   thing_name your-iot-thing-name
   ```

L'`shadow.py`exemple d'application s'exécute ensuite et récupère l'état d'ombre actuel. Si vous avez supprimé l'ombre ou effacé les états actuels, le programme définit la valeur actuelle sur `off` puis vous invite à en saisir une `desired`.

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to Delta events...
Subscribing to Update responses...
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow document lacks 'color' property. Setting defaults...
Changed local shadow value to 'off'.
Updating reported shadow value to 'off'...
Update request published.
Finished updating reported shadow value to 'off'...
Enter desired value:
```

En revanche, si le programme était en cours d'exécution et que vous l'avez redémarré, vous verrez la dernière valeur de couleur signalée dans le terminal. Dans le client de test MQTT, vous verrez une mise à jour des rubriques $aws/things/ /shadow/get et $aws/things//***thingname***. ***thingname*** shadow/get/accepted

Supposons que la dernière couleur signalée soit `green`. Ce qui suit montre le contenu du fichier $aws/things//JSON ***thingname***. shadow/get/accepted

```
{
"state": {
  "desired": {
    "welcome": "aws-iot",
    "color": "green"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "green"
  }
},
"metadata": {
  "desired": {
```

```
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  }
},
"version": 10,
"timestamp": 1620173908
}
```

b. Entrez une `desired` valeur dans le terminal, telle que`yellow`. L'`shadow.py`exemple d'application répond et affiche les messages suivants dans le terminal qui indiquent la modification de la `reported` valeur de`yellow`.

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

Dans le client de test MQTT de la AWS IoT console, sous Abonnements, vous pouvez voir que les sujets suivants ont reçu un message :

- $aws/things/ ***`thingname`*** /shadow/update : indique que les deux `desired` valeurs et les valeurs changent de couleur. `updated` `yellow`
- $aws/things/***`thingname`***/shadow/update/accepted: affiche les valeurs actuelles des `reported` états `desired` et ainsi que leurs métadonnées et informations de version.

- $aws/things/***thingname***/shadow/update/documents: affiche les valeurs précédentes et actuelles des `reported` états et ainsi que leurs métadonnées `desired` et informations de version.

Comme le document $aws/things/***thingname***/contient shadow/update/documents également des informations contenues dans les deux autres rubriques, nous pouvons le consulter pour voir les informations d'état. L'état précédent indique la valeur signalée définie sur`green`, ses métadonnées et ses informations de version, ainsi que l'état actuel qui indique la valeur signalée mise à jour vers `yellow`.

```
{
"previous": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "green"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "green"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    }
  },
  "version": 10
```

```
},
"current": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "yellow"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "yellow"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    }
  },
  "version": 11
},
"timestamp": 1617297904
}
```

c. Désormais, si vous entrez une autre `desired` valeur, d'autres modifications sont apportées aux `reported` valeurs et des mises à jour des messages reçues par ces rubriques. Le numéro de version est également incrémenté de 1. Par exemple, si vous entrez la valeur `green`, l'état précédent indique la valeur `yellow` et l'état actuel indique la valeur`green`.

4. Modifier le document Shadow pour observer les événements du delta

   Pour observer les modifications apportées à la rubrique delta, modifiez le document Shadow dans la AWS IoT console. Par exemple, vous pouvez remplacer la `desired` valeur par la

couleur`red`. Pour ce faire, dans la AWS IoT console, choisissez Modifier, puis définissez la `desired` valeur en rouge dans le JSON, tout en conservant la `reported` valeur définie sur`green`. Avant d'enregistrer la modification, laissez le terminal ouvert car vous verrez le message delta s'afficher dans le terminal.

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

L'`shadow.py`exemple d'application répond à cette modification et affiche des messages dans le terminal indiquant le delta. Dans le client de test MQTT, les `update` sujets auront reçu un message indiquant les modifications apportées aux valeurs `desired` et `reported`.

Vous voyez également que le sujet $aws/things/***thingname***/shadow/update/deltaa reçu un message. Pour voir le message, choisissez ce sujet, qui est répertorié sous Abonnements.

```
{
"version": 13,
"timestamp": 1617318480,
"state": {
  "color": "red"
},
"metadata": {
  "color": {
    "timestamp": 1617318480
  }
}
}
```

Étape 3 : Résolution des erreurs liées aux interactions entre Device Shadow et Device Shadow

Lorsque vous exécutez l'application Shadow sample, vous pouvez rencontrer des problèmes lors de l'observation des interactions avec le service Device Shadow.

Si le programme s'exécute correctement et vous invite à saisir une `desired` valeur, vous devriez pouvoir observer les interactions entre Device Shadow en utilisant le document Shadow et le client de test MQTT comme décrit précédemment. Toutefois, si vous ne parvenez pas à voir les interactions, voici quelques points que vous pouvez vérifier :

- Vérifiez le nom de l'objet et son ombre dans la AWS IoT console

  Si vous ne voyez pas les messages dans le document Shadow, vérifiez la commande et assurez-vous qu'elle correspond au nom de l'objet dans la AWS IoT console. Vous pouvez également vérifier si vous avez une ombre classique en choisissant votre ressource matérielle, puis en choisissant Shadows. Ce tutoriel se concentre principalement sur les interactions avec l'ombre classique.

  Vous pouvez également vérifier que l'appareil que vous avez utilisé est connecté à Internet. Dans la AWS IoT console, choisissez l'élément que vous avez créé précédemment, puis choisissez Interact. Sur la page des détails de l'objet, vous devriez voir un message disant : `This thing already appears to be connected.`

- Vérifiez les sujets réservés au MQTT auxquels vous vous êtes abonné

  Si les messages ne s'affichent pas dans le client de test MQTT, vérifiez si les sujets auxquels vous vous êtes abonné sont correctement formatés. Les sujets MQTT Device Shadow ont le format $aws/things/ ***`thingname`*** /shadow/ et peuvent avoir `update` ou `delete` suivre ce format en fonction des actions que vous souhaitez effectuer sur l'ombre. `get` Ce didacticiel utilise le sujet $aws/things/ ***`thingname`*** /shadow/ #. Assurez-vous donc de l'avoir saisi correctement lorsque vous vous êtes abonné au sujet dans la section Filtre de sujets du client de test.

  Lorsque vous entrez le nom du sujet, assurez-vous qu'il *`thingname`* est identique au nom de l' AWS IoT élément que vous avez créé précédemment. Vous pouvez également vous abonner à des rubriques MQTT supplémentaires pour voir si une mise à jour a été effectuée avec succès. Par exemple, vous pouvez vous abonner à la rubrique $aws/things/***`thingname`***/shadow/update/rejectedpour recevoir un message chaque fois qu'une demande de mise à jour échoue afin de pouvoir résoudre les problèmes de connexion. Pour de plus amples informations sur les rubriques réservées pour les shadows, veuillez consulter the section called “Rubriques de shadow” et Rubriques MQTT de Device Shadow.

Étape 4 : examen des résultats et des étapes suivantes

Dans ce didacticiel, vous allez découvrir comment :

- Utilisez l'`shadow.py`exemple d'application pour spécifier les états souhaités et mettre à jour l'état actuel de l'ombre.
- Modifiez le document Shadow pour observer les événements delta et la façon dont l'`shadow.py`exemple d'application y répond.
- Utilisez le client de test MQTT pour vous abonner à des sujets fictifs et observer les mises à jour lorsque vous exécutez l'exemple de programme.

Étapes suivantes

Vous pouvez vous abonner à d'autres rubriques réservées au MQTT pour suivre les mises à jour de l'application parallèle. Par exemple, si vous vous abonnez uniquement à la rubrique $aws/things/***thingname***/shadow/update/accepted, vous ne verrez que les informations d'état actuel lorsqu'une mise à jour est effectuée avec succès.

Vous pouvez également vous abonner à des rubriques supplémentaires pour résoudre les problèmes ou en savoir plus sur les interactions entre Device Shadow et également résoudre les problèmes liés aux interactions Device Shadow. Pour plus d'informations, consultez the section called "Rubriques de shadow" et Rubriques MQTT de Device Shadow.

Vous pouvez également choisir d'étendre votre application en utilisant des ombres nommées ou en utilisant du matériel supplémentaire connecté au Raspberry Pi pour observer LEDs les modifications de leur état à l'aide des messages envoyés depuis le terminal.

Pour plus d'informations sur le service Device Shadow et son utilisation dans les appareils, les applications et les servicesAWS IoT Service Device Shadow, consultezUtilisation des shadows sur les appareils, etUtilisation des shadows dans les applications et les services.

Didacticiel : Création d'un mécanisme d'autorisation personnalisé pour AWS IoT Core

Ce didacticiel explique les étapes de création, de validation et d'utilisation de l'authentification personnalisée à l'aide du AWS CLI. En option, à l'aide de ce didacticiel, vous pouvez utiliser Postman pour envoyer des données à l'aide AWS IoT Core de la fonction HTTP PublierAPI.

Ce didacticiel explique comment créer un exemple de fonction Lambda qui implémente la logique d'autorisation et d'authentification et un mécanisme d'autorisation à l'aide de l'appel create-authorizer avec signature par jeton activée. L'autorisateur est ensuite validé à l'aide dutest-invoke-authorizer, et vous pouvez enfin envoyer des données à AWS IoT Core en utilisant le bouton HTTP API Publier sur un MQTT sujet de test. Un exemple de demande indiquera l'autorisateur à invoquer en utilisant l'`x-amz-customauthorizer-name`en-tête et en transmettant les en-têtes de demande token-key-name et `x-amz-customauthorizer-signature` in.

Ce que vous allez apprendre dans ce didacticiel:

- Comment créer une fonction Lambda en tant que gestionnaire de mécanisme d'autorisation personnalisé
- Comment créer un autorisateur personnalisé à l'aide du bouton AWS CLI avec signature par jeton activée
- Comment tester votre mécanisme d'autorisation personnalisé à l'aide de la commande test-invoke-authorizer
- Comment publier un MQTT sujet à l'aide de Postman et valider la demande avec votre autorisateur personnalisé

Ce didacticiel vous prendra environ 60 minutes.

Dans ce tutoriel, vous allez :

- Étape 1 : Créez une fonction Lambda pour votre mécanisme d'autorisation personnalisé
- Étape 2 : Créez une paire de clés publique et privée pour votre mécanisme d'autorisation personnalisé
- Étape 3 : créer une ressource d'autorisation personnalisée et son autorisation
- Étape 4 : Testez l'autorisateur en appelant test-invoke-authorizer
- Étape 5 : Tester la publication d'MQTTun message à l'aide de Postman
- Étape 6 : Afficher les messages dans le client MQTT de test
- Étape 7 : examen des résultats et des étapes suivantes
- Étape 8 : Nettoyer

Avant de commencer ce didacticiel, assurez-vous de disposer des éléments suivants :

- Configurez Compte AWS

Vous aurez besoin de votre AWS IoT console Compte AWS et de votre console pour terminer ce didacticiel.

Le compte que vous utilisez pour ce didacticiel fonctionne mieux lorsqu'il inclut au moins les politiques AWS gérées suivantes :

- `IAMFullAccess`
- `AWSIoTFullAccess`
- `AWSLambda_FullAccess`

> **Important**
>
> Les IAM politiques utilisées dans ce didacticiel sont plus permissives que celles que vous devriez suivre dans le cadre d'une implémentation de production. Dans un environnement de production, assurez-vous que vos politiques de compte et de ressources n'accordent que les autorisations nécessaires.
> Lorsque vous créez des IAM politiques pour la production, déterminez l'accès dont les utilisateurs et les rôles ont besoin, puis concevez les politiques qui leur permettent d'effectuer uniquement ces tâches.
> Pour plus d'informations, consultez la section Bonnes pratiques en matière de sécurité dans IAM

- A installé le AWS CLI

  Pour plus d'informations sur l'installation du AWS CLI, reportez-vous à la section Installation du AWS CLI. Ce didacticiel nécessite une AWS CLI version `aws-cli/2.1.3 Python/3.7.4 Darwin/18.7.0 exe/x86_64` ou une version ultérieure.

- SSLOutils ouverts

  Les exemples de ce didacticiel utilisent Libre SSL 2.6.5. Vous pouvez également utiliser les outils Open SSL v1.1.1i pour ce didacticiel.

- J'ai revu l'AWS Lambda aperçu

  Si vous ne l'avez jamais utilisé AWS Lambda auparavant, consultez le AWS Lambdaguide Getting started with Lambda pour en apprendre les termes et les concepts.

- J'ai examiné comment créer des demandes dans Postman

  Pour plus d'informations, consultez Demandes de construction.

- Autorisateurs personnalisés supprimés du didacticiel précédent

  Vous ne Compte AWS pouvez configurer qu'un nombre limité d'autorisateurs personnalisés à la fois. Pour plus d'informations sur la suppression d'un mécanisme d'autorisation personnalisé, consultez the section called “Étape 8 : Nettoyer”.

Étape 1 : Créez une fonction Lambda pour votre mécanisme d'autorisation personnalisé

L'authentification personnalisée AWS IoT Core utilise les ressources d'autorisation que vous créez pour authentifier et autoriser les clients. La fonction que vous allez créer dans cette section authentifie et autorise les clients lorsqu'ils se connectent aux ressources AWS IoT Core et y accèdent AWS IoT .

La fonction Lambda effectue les opérations suivantes :

- Si une demande provient detest-invoke-authorizer, elle renvoie une IAM politique avec une `Deny` action.
- Si une demande provient de Postman using HTTP et que le `actionToken` paramètre a une valeur de`allow`, il renvoie une IAM politique avec une `Allow` action. Dans le cas contraire, il renvoie une IAM politique avec une `Deny` action.

Pour créer une fonction Lambda pour votre mécanisme d'autorisation personnalisé

1. Dans la console Lambda, ouvrez Fonctions.
2. Sélectionnez Create function (Créer une fonction).
3. Confirmez que l'auteur à partir de zéro (Author from scratch) est sélectionné.
4. Sous Basic information :

   a. Sous Nom de la fonction, entrez **`custom-auth-function`**.

   b. sSous Runtime, confirmez Node.js 18.x
5. Sélectionnez Create function (Créer une fonction).

   Lambda crée une fonction Node.js et un rôle d'exécution qui accorde à la fonction l'autorisation de charger des journaux. La fonction Lambda assume le rôle d'exécution lorsque vous appelez votre fonction et utilise le rôle d'exécution pour créer des informations d'identification AWS SDK et pour lire les données des sources d'événements.

6. Pour voir le code et la configuration de la fonction dans l'AWS Cloud9éditeur, choisissez custom-auth-functiondans la fenêtre du concepteur, puis choisissez index.js dans le volet de navigation de l'éditeur.

   Pour créer une fonction Lambda, sélectionnez l'Amazon Resource Name (ARN) de la fonction Lambda. Node.js Vous pouvez utiliser l'éditeur AWS Cloud9 pour éditer votre fonction tant que votre code source ne dépasse pas 3 Mo.

7. Remplacez le code index.js dans l'éditeur par le code suivant :

```
// A simple Lambda function for an authorizer. It demonstrates
// How to parse a CLI and Http password to generate a response.

export const handler = async (event, context, callback) => {

    //Http parameter to initiate allow/deny request
    const HTTP_PARAM_NAME='actionToken';
    const ALLOW_ACTION = 'Allow';
    const DENY_ACTION = 'Deny';

    //Event data passed to Lambda function
    var event_str = JSON.stringify(event);
    console.log('Complete event :'+ event_str);

    //Read protocolData from the event json passed to Lambda function
    var protocolData = event.protocolData;
    console.log('protocolData value---> ' + protocolData);

    //Get the dynamic account ID from function's ARN to be used
    // as full resource for IAM policy
    var ACCOUNT_ID = context.invokedFunctionArn.split(":")[4];
    console.log("ACCOUNT_ID---"+ACCOUNT_ID);

    //Get the dynamic region from function's ARN to be used
    // as full resource for IAM policy
    var REGION = context.invokedFunctionArn.split(":")[3];
    console.log("REGION---"+REGION);

    //protocolData data will be undefined if testing is done via CLI.
    // This will help to test the set up.
    if (protocolData === undefined) {

        //If CLI testing, pass deny action as this is for testing purpose only.
```

```
        console.log('Using the test-invoke-authorizer cli for testing only');
        callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

    } else{

        //Http Testing from Postman
        //Get the query string from the request
        var queryString = event.protocolData.http.queryString;
        console.log('queryString values -- ' + queryString);
        /*         global URLSearchParams       */
        const params = new URLSearchParams(queryString);
        var action = params.get(HTTP_PARAM_NAME);

        if(action!=null && action.toLowerCase() === 'allow'){

            callback(null, generateAuthResponse(ALLOW_ACTION,ACCOUNT_ID,REGION));

        }else{

            callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

        }

    }

};

// Helper function to generate the authorization IAM response.
var generateAuthResponse = function(effect,ACCOUNT_ID,REGION) {

    var full_resource = "arn:aws:iot:"+ REGION + ":" + ACCOUNT_ID + ":*";
    console.log("full_resource---"+full_resource);

    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'principalId';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var statement = {};
    statement.Action = 'iot:*';
    statement.Effect = effect;
    statement.Resource = full_resource;
```

```
    policyDocument.Statement[0] = statement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 600;

    console.log('custom auth policy function called from http');
    console.log('authResponse --> ' + JSON.stringify(authResponse));
    console.log(authResponse.policyDocuments[0]);

    return authResponse;
}
```

8. Choisissez Deploy (Déployer).
9. Après le déploiement des modifications (Changes deployed) apparaît au-dessus de l'éditeur :
   a. Accédez à la section Vue d'ensemble des fonctions au-dessus de l'éditeur.
   b. Copiez la fonction ARN et enregistrez-la pour l'utiliser ultérieurement dans ce didacticiel.
10. Testez votre fonction .
    a. Choisissez l’onglet Test.
    b. À l'aide des paramètres de test par défaut, choisissez Invoquer.
    c. Si le test a réussi, dans les résultats de l'exécution, ouvrez l'aperçu Détails. Vous devriez voir le document de politique renvoyé par la fonction.

       Si le test a échoué ou si aucun document de politique ne s'affiche, examinez le code pour rechercher et corriger les erreurs.

Étape 2 : Créez une paire de clés publique et privée pour votre mécanisme d’autorisation personnalisé

Votre mécanisme d'autorisation personnalisé nécessite une clé publique et privée pour l'authentifier. Les commandes de cette section utilisent les SSL outils Open pour créer cette paire de clés.

Pour créer une paire de clés publique et privée pour votre mécanisme d’autorisation personnalisé

1. Créez le fichier de clé privée.

```
openssl genrsa -out private-key.pem 4096
```

2. Vérifiez le fichier de clé privée que vous venez de créer.

```
openssl rsa -check -in private-key.pem -noout
```

Si la commande n'affiche aucune erreur, le fichier de clé privée est valide.

3. Créez le fichier de clé publique.

```
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

4. Vérifiez le fichier de clé publique.

```
openssl pkey -inform PEM -pubin -in public-key.pem -noout
```

Si la commande n'affiche aucune erreur, le fichier de clé publique est valide.

Étape 3 : créer une ressource d'autorisation personnalisée et son autorisation

L'autorisateur AWS IoT personnalisé est la ressource qui réunit tous les éléments créés au cours des étapes précédentes. Dans cette section, vous allez créer une ressource de mécanisme d'autorisation personnalisée et lui donner l'autorisation d'exécuter la fonction Lambda que vous avez créée précédemment. Vous pouvez créer une ressource d'autorisation personnalisée à l'aide de la AWS IoT console, du AWS CLI, ou du AWS API.

Pour ce didacticiel, il vous suffit de créer un seul mécanisme d'autorisation personnalisé. Cette section décrit comment créer à l'aide de la AWS IoT console et du AWS CLI, afin que vous puissiez utiliser la méthode qui vous convient le mieux. Il n'y a aucune différence entre les ressources de mécanisme d'autorisation personnalisées créées par l'une ou l'autre méthode.

Création d'une ressource d'autorisation personnalisée

Choisissez l'une de ces options pour créer votre ressource de mécanisme d'autorisation personnalisée

- Créez un autorisateur personnalisé à l'aide de la console AWS IoT
- Créer un mécanisme d'autorisation à l'aide de AWS CLI

Pour créer un mécanisme d'autorisation personnalisé (console)

1. Ouvrez la page d'autorisation personnalisée de la AWS IoT console, puis choisissez Create Authorizer.
2. Dans Créer un mécanisme d'autorisation
   a. Dans Nom de mécanisme d'autorisation, entrez **`my-new-authorizer`**.
   b. Dans État du mécanisme d'autorisation, cochez Actif.
   c. Dans Fonction mécanisme d'autorisation, choisissez la fonction Lambda que vous avez créée précédemment.
   d. Dans Validation du jeton, facultatif :
      i. Activez la validation des jetons.
      ii. Dans Nom de la clé du jeton, entrez**`tokenKeyName`**.
      iii. Sélectionnez Ajouter une clé.
      iv. Dans Key name (Nom de la clé), saisissez **`FirstKey`**.
      v. Dans Clé publique, entrez le contenu du fichier `public-key.pem`. Veillez à inclure les lignes du fichier dans le contenu du fichier `-----BEGIN PUBLIC KEY-----` et `-----END PUBLIC KEY-----` à ne pas ajouter ou supprimer des fils de ligne, des retours en chariot ou d'autres caractères. La chaîne que vous entrez devrait ressembler à cet exemple.

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----
```

3. Choisissez Créer un mécanisme d'autorisation.

4. Si la ressource du mécanisme d'autorisation personnalisée a été créée, vous verrez la liste des mécanismes d'autorisation personnalisés et votre nouvel mécanisme d'autorisation personnalisé devrait apparaître dans la liste. Vous pouvez passer à la section suivante pour le tester.

   Si une erreur s'affiche, examinez-la, réessayez de créer votre mécanisme d'autorisation personnalisé et vérifiez les entrées. Notez que chaque ressource de mécanisme d'autorisation personnalisée doit avoir un nom unique.

Pour créer un mécanisme d'autorisation personnalisé (AWS CLI)

1. Remplacez vos valeurs par `authorizer-function-arn` et `token-signing-public-keys`, puis exécutez la commande suivante :

```
aws iot create-authorizer \
--authorizer-name "my-new-authorizer" \
--token-key-name "tokenKeyName" \
--status ACTIVE \
--no-signing-disabled \
--authorizer-function-arn "arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function" \
--token-signing-public-keys FirstKey="-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----"
```

   Où :

   - La `authorizer-function-arn` valeur est le nom de ressource Amazon (ARN) de la fonction Lambda que vous avez créée pour votre autorisateur personnalisé.

- La valeur `token-signing-public-keys` inclut le nom de la clé **`FirstKey`** et le contenu du fichier `public-key.pem`. Veillez à inclure les lignes du fichier dans le contenu du fichier `-----BEGIN PUBLIC KEY-----` et `-----END PUBLIC KEY-----` à ne pas ajouter ou supprimer des fils de ligne, des retours en chariot ou d'autres caractères.

  Remarque : soyez prudent en saisissant la clé publique car toute modification de la valeur de la clé publique la rend inutilisable.

2. Si l'autorisateur personnalisé est créé, la commande renvoie le nom et ARN la nouvelle ressource, tels que les suivants.

```
{
    "authorizerName": "my-new-authorizer",
    "authorizerArn": "arn:aws:iot:Region:57EXAMPLE833:authorizer/my-new-authorizer"
}
```

   Notez la sortie `authorizerArn` pour l'utiliser lors de l'étape suivante.

   N'oubliez pas que chaque ressource de mécanisme d'autorisation personnalisée doit avoir un nom unique.

Autoriser la ressource de mécanisme d'autorisation personnalisée

Dans cette section, vous accorderez à la ressource d'autorisation personnalisée que vous venez de créer l'autorisation d'exécuter la fonction Lambda. Pour accorder l'autorisation, vous pouvez utiliser la CLI commande add permission.

Accordez l'autorisation à votre fonction Lambda à l'aide du AWS CLI

1. Après avoir inséré vos valeurs, entrez la commande suivante. Notez que la valeur `statement-id` doit être unique. Remplacez *`Id-1234`* par une autre valeur si vous avez déjà exécuté ce didacticiel ou si une erreur `ResourceConflictException` s'affiche.

```
aws lambda add-permission  \
--function-name "custom-auth-function" \
--principal "iot.amazonaws.com" \
--action "lambda:InvokeFunction" \
--statement-id "Id-1234" \
--source-arn authorizerArn
```

2. Si la commande aboutit, elle renvoie une instruction d'autorisation, comme dans cet exemple. Vous pouvez passer à la section suivante pour tester le mécanisme d'autorisation personnalisée.

```
{
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
\":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
\",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
}
```

Si la commande échoue, elle renvoie une erreur, comme dans cet exemple. Vous devez vérifier et corriger l'erreur avant de continuer.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Étape 4 : Testez l'autorisateur en appelant test-invoke-authorizer

Une fois toutes les ressources définies, dans cette section, vous appellerez test-invoke-authorizer depuis la ligne de commande pour tester la passe d'autorisation.

Notez que lorsque vous invoquez l'autorisateur depuis la ligne de commande, `protocolData` il n'est pas défini, de sorte que l'autorisateur renvoie toujours un document. DENY Ce test confirme toutefois que votre mécanisme d'autorisation personnalisé et votre fonction Lambda sont correctement configurés, même s'ils ne testent pas complètement la fonction Lambda.

Pour tester votre autorisateur personnalisé et sa fonction Lambda à l'aide du AWS CLI

1. Dans le répertoire où vous avez créé le fichier `private-key.pem`, entrez la commande suivante.

```
echo -n "tokenKeyValue" | openssl dgst -sha256 -sign private-key.pem | openssl
 base64 -A
```

Cette commande crée une chaîne de signature à utiliser à l'étape suivante. La chaîne de signature ressemble à ceci :

```
dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V
+MMWu09YSA86+64Y3Gt4tOykpZqn9mn
VB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeeh
bQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnuOBvMGCEFE09jGjj
szEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY
+d1vTvdthKtYHBq8MjhzJ0kggbt29V
QJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuX
f3LzCwQQF/YSUy02u5XkWn
+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+K
EWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFH
xRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

Copiez cette chaîne de signature pour l'utiliser à l'étape suivante. Veillez à ne pas inclure de caractères supplémentaires ou à ne pas en omettre.

2. Dans cette commande, remplacez la valeur `token-signature` par la chaîne de signature de l'étape précédente et exécutez cette commande pour tester votre mécanisme d’autorisation.

```
aws iot test-invoke-authorizer \
--authorizer-name my-new-authorizer \
--token tokenKeyValue \
--token-signature dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr
+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp
+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsH
+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/
P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuXf3LzCwQQF/YSUy02u5XkWn
+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o
+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

Si la commande aboutit, elle renvoie les informations générées par votre fonction d'autorisation personnalisée, comme dans cet exemple.

```
{
    "isAuthenticated": true,
    "principalId": "principalId",
    "policyDocuments": [
        "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Action\":\"iot:*\",\"Effect
\":\"Deny\",\"Resource\":\"arn:aws:iot:Region:57EXAMPLE833:*\"}]}"
    ],
    "refreshAfterInSeconds": 600,
    "disconnectAfterInSeconds": 3600
}
```

Si la commande renvoie une erreur, vérifiez l'erreur et revérifiez les commandes que vous avez utilisées dans cette section.

Étape 5 : Tester la publication d'MQTTun message à l'aide de Postman

1. Pour obtenir le point de terminaison des données de votre appareil depuis la ligne de commande, appelez describe-endpoint comme indiqué ici

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

   Enregistrez cette adresse pour une utilisation ultérieure. *device_data_endpoint_address*
2. Ouvrez une nouvelle fenêtre Postman et créez une nouvelle HTTP POST demande.

   a. Sur votre ordinateur, ouvrez l'application Postman.

   b. Dans Postman, dans le menu File (Fichier), choisissez New… (Nouveau).

   c. Dans la boîte de dialogue Nouveau, choisissez Requête.

   d. Dans Enregistrer la requête,

      i. Dans Nom de la requête, entrez **Custom authorizer test request**.

      ii. Dans Sélectionner une collection ou un dossier dans lequel enregistrer : choisissez ou créez une collection dans laquelle enregistrer cette demande.

      iii. Choisissez Enregistrer dans ***collection_name***.
3. Créez la POST demande pour tester votre autorisateur personnalisé.

   a. Dans le sélecteur de méthode de demande situé à côté du URL champ, sélectionnez POST.

b. Dans le URL champ, créez le URL pour votre demande en utilisant ce qui suit URL avec la commande *device_data_endpoint_address* from the describe-endpoint lors d'une étape précédente.

```
https://device_data_endpoint_address:443/topics/test/cust-auth/topic?
qos=0&actionToken=allow
```

Notez que cela URL inclut le paramètre de `actionToken=allow` requête qui indiquera à votre fonction Lambda de renvoyer un document de politique autorisant l'accès à. AWS IoT Une fois que vous avez saisi leURL, les paramètres de requête apparaissent également dans l'onglet Paramètres de Postman.

c. Dans l'onglet Auth, dans le champ Type, sélectionnez No Auth.

d. Dans la section suivante, passez à l'étape suivante.

i. Si une clé d'hôte est cochée, décochez-la.

ii. Au bas de la liste des en-têtes, ajoutez ces nouveaux en-têtes et confirmez qu'ils sont cochés. Remplacez la **Host** valeur par votre *device_data_endpoint_address* et la **x-amz-customauthorizer-signature** valeur par la chaîne de signature que vous avez utilisée avec la test-invoke-authorize commande de la section précédente.

Clé	Valeur
x-amz-customauthorizer-name	**my-new-authorizer**
Host	*device_data_endpoint_address*
tokenKeyName	**tokenKeyValue**

Clé	Valeur
x-amz-customauthorizer-signature	*dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnu0BvMGCEFE09jGjjszEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca+tsDuXf3LzCwQQF/YSUy02u5XkWn+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=*

e. Dans l'onglet Body :

i. Dans la zone d'option du format des données, choisissez Raw.

ii. Dans la liste des types de données, sélectionnez JavaScript.

iii. Dans le champ de texte, entrez la charge utile suivante pour votre message de test : JSON

```
{
```

```
    "data_mode": "test",
    "vibration": 200,
    "temperature": 40
}
```

4. Choisissez Send pour envoyer la requête.

   Si la requête aboutit, elle renvoie :

   ```
   {
       "message": "OK",
       "traceId": "ff35c33f-409a-ea90-b06f-fbEXAMPLE25c"
   }
   ```

   La réponse positive indique que votre autorisateur personnalisé a autorisé la connexion à AWS IoT et que le message de test a été remis au broker in AWS IoT Core.

   S'il renvoie une erreur, consultez le message d'erreur*device_data_endpoint_address*, la chaîne de signature et les autres valeurs d'en-tête.

Conservez cette demande dans Postman pour l'utiliser dans la section suivante.

Étape 6 : Afficher les messages dans le client MQTT de test

À l'étape précédente, vous avez envoyé des messages simulés à un appareil à l'aide AWS IoT de Postman. La réponse positive indique que votre mécanisme d’autorisation personnalisé a autorisé la connexion à AWS IoT et que le message de test a été livré au courtier en AWS IoT Core. Dans cette section, vous allez utiliser le client de MQTT test de la AWS IoT console pour voir le contenu du message, comme le feraient d'autres appareils et services.

Pour voir les messages de test autorisés par votre mécanisme d’autorisation personnalisé

1. Dans la AWS IoT console, ouvrez le client MQTT de test.
2. Dans l'onglet S'abonner à la rubrique, dans le filtre de rubrique**`test/cust-auth/topic`**, entrez la rubrique du message utilisé dans l'exemple Postman de la section précédente.
3. Choisissez Souscrire.

   Gardez cette fenêtre visible pour l'étape suivante.

4. Dans Postman, dans la demande que vous avez créée pour la section précédente, choisissez Envoyer.

   Passez en revue la réponse pour vous assurer qu'elle a bien été prise en compte. Si ce n'est pas le cas, corrigez l'erreur comme décrit dans la section précédente.

5. Dans le client de MQTT test, vous devriez voir une nouvelle entrée indiquant le sujet du message et, s'il est développé, la charge utile du message provenant de la demande que vous avez envoyée depuis Postman.

   Si vos messages ne s'affichent pas dans le client de MQTT test, voici quelques points à vérifier :

   - Assurez-vous que votre demande Postman a été renvoyée avec succès. En cas de AWS IoT rejet de la connexion et de renvoi d'une erreur, le message contenu dans la demande n'est pas transmis au courtier de messages.
   - Assurez-vous que le Compte AWS et Région AWS utilisé pour ouvrir la AWS IoT console sont les mêmes que ceux que vous utilisez dans le PostmanURL.
   - Assurez-vous que vous utilisez le point de terminaison approprié pour l'autorisateur personnalisé. Le point de terminaison IoT par défaut peut ne pas prendre en charge l'utilisation d'autorisateurs personnalisés dotés de fonctions Lambda. Vous pouvez plutôt utiliser des configurations de domaine pour définir un nouveau point de terminaison, puis spécifier ce point de terminaison pour l'autorisateur personnalisé.
   - Assurez-vous d'avoir correctement saisi le sujet dans le client de MQTT test. Le filtre de sujet est sensible à la casse. En cas de doute, vous pouvez également vous abonner au **#** sujet, qui s'abonne à tous les MQTT messages qui passent par le courtier de messages Compte AWS et Région AWS utilisés pour ouvrir la AWS IoT console.

Étape 7 : examen des résultats et des étapes suivantes

Dans ce tutoriel :

- Vous avez créé une fonction Lambda pour être un gestionnaire de mécanisme d'autorisation personnalisé
- Vous avez créé un mécanisme d'autorisation personnalisé avec la signature par jeton activée
- Vous avez testé votre mécanisme d'autorisation personnalisé à l'aide de la commande test-invoke-authorizer

- Vous avez publié un MQTT sujet à l'aide de Postman et validez la demande avec votre autorisateur personnalisé
- Vous avez utilisé le client de MQTT test pour afficher les messages envoyés depuis votre test Postman

Étapes suivantes

Après avoir envoyé des messages de Postman pour vérifier que le mécanisme d'autorisation personnalisé fonctionne, essayez d'expérimenter pour voir comment la modification des différents aspects de ce didacticiel affecte les résultats. Voici quelques exemples pour vous aider à démarrer.

- Modifiez la chaîne de signature afin qu'elle ne soit plus valide pour voir comment les tentatives de connexion non autorisées sont traitées. Vous devriez recevoir une réponse d'erreur, comme celle-ci, et le message ne devrait pas apparaître dans le client de MQTT test.

```
{
    "message": "Forbidden",
    "traceId": "15969756-a4a4-917c-b47a-5433e25b1356"
}
```

- Pour en savoir plus sur la détection des erreurs susceptibles de se produire lors du développement et de l'utilisation de AWS IoT règles, consultezSurveillance AWS IoT.

Étape 8 : Nettoyer

Si vous souhaitez répéter ce didacticiel, vous devrez peut-être supprimer certains de vos mécanismes d'autorisation personnalisés. Vous ne Compte AWS pouvez configurer qu'un nombre limité d'autorisateurs personnalisés à la fois et vous pouvez en obtenir un `LimitExceededException` lorsque vous essayez d'en ajouter un nouveau sans supprimer un autorisateur personnalisé existant.

Pour supprimer un mécanisme d'autorisation personnalisé (console)

1. Ouvrez la page d'autorisation personnalisée de la AWS IoT console et, dans la liste des autorisateurs personnalisés, recherchez l'autorisateur personnalisé à supprimer.
2. Ouvrez la page de détails de mécanisme d'autorisation personnalisé et, dans le menu Actions, choisissez Modifier.
3. Décochez la case Activer l'autorisateur, puis choisissez Mettre à jour.

Vous ne pouvez pas supprimer un mécanisme d'autorisation personnalisé lorsqu'il est actif.

4. Sur la page Détails du mécanisme d’autorisation personnalisé, ouvrez le menu Actions et choisissez Supprimer.

Pour supprimer un mécanisme d'autorisation personnalisé (AWS CLI)

1. Répertoriez les mécanismes d’autorisation personnalisées que vous avez installées et recherchez ceux que vous souhaitez supprimer.

```
aws iot list-authorizers
```

2. Définissez le mécanisme d'autorisation personnalisé sur `inactive` en exécutant cette commande après avoir remplacé *`Custom_Auth_Name`* par `authorizerName` du mécanisme d'autorisation personnalisé à supprimer.

```
aws iot update-authorizer --status INACTIVE --authorizer-name Custom_Auth_Name
```

3. Supprimez le mécanisme d'autorisation personnalisé *`Custom_Auth_Name`* en exécutant cette commande après avoir remplacé `authorizerName` par du mécanisme d'autorisation personnalisé à supprimer.

```
aws iot delete-authorizer --authorizer-name Custom_Auth_Name
```

Tutoriel : Surveillance de l'humidité du sol avec un AWS IoT Raspberry Pi

Ce didacticiel vous explique comment utiliser un Raspberry Pi, un capteur d'humidité, et comment AWS IoT surveiller le niveau d'humidité du sol pour une plante d'intérieur ou un jardin. Le Raspberry Pi exécute un code qui lit le niveau d'humidité et la température à partir du capteur, puis envoie les données à AWS IoT. Vous créez une règle AWS IoT qui envoie un e-mail à une adresse abonnée à une rubrique Amazon SNS lorsque le niveau d'humidité tombe en dessous d'un seuil.

> **Note**
>
> Ce didacticiel n'est peut-être pas à jour. Certaines références ont peut-être été remplacées depuis la publication initiale de ce sujet.

Table des matières

- Prérequis
- Con AWS IoT figuration
  - Étape 1 : Création de la AWS IoT politique
  - Étape 2 : Création de l' AWS IoT objet, du certificat et de la clé privée
  - 3e étape : Créer une rubrique Amazon SNS et s'abonner
  - Étape 4 : créer une AWS IoT règle pour envoyer un e-mail
- Configuration de votre Raspberry Pi et du capteur d'humidité

Prérequis

Pour suivre ce didacticiel, vous devez disposer des éléments suivants :

- Un Compte AWS.
- Utilisateur IAM possédant des autorisations de niveau administrateur.
- Un ordinateur de développement exécutant Windows, macOS, Linux ou Unix pour accéder à la console AWS IoT.
- Un Raspberry Pi 3B ou 4B exécutant le dernier système d'exploitation Raspberry Pi. Pour les instructions d'installation, voir Installer un système d'exploitation sur le site Web du Raspberry Pi.
- Un écran, un clavier, une souris et un réseau Wi-Fi ou une connexion Ethernet pour votre Raspberry Pi.
- Un capteur d'humidité compatible avec Raspberry Pi. Le capteur utilisé dans ce didacticiel est un capteur d'humidité capacitif SteMMA I2C Adafruit avec un en-tête de câble à 4 broches vers connecteur femelle JST.

Con AWS IoT figuration

Pour suivre ce didacticiel, vous devez créer les ressources suivantes. Pour connecter un appareil à AWS IoT, vous devez créer un objet IoT, un certificat d'appareil et une AWS IoT politique.

- N' AWS IoT importe quoi.

  Un objet représente un appareil physique (dans ce cas, votre Rasberry Pi) et contient des métadonnées statiques sur l'appareil.
- Un certificat d'appareil.

Tous les appareils doivent avoir un certificat d'appareil pour se connecter à AWS IoT et s'authentifier auprès de celui-ci.

- Une AWS IoT politique.

  Une ou plusieurs AWS IoT politiques sont associées à chaque certificat d'appareil. Ces politiques déterminent les AWS IoT ressources auxquelles l'appareil peut accéder.

- Un certificat CA AWS IoT racine.

  Les appareils et autres clients utilisent un certificat CA AWS IoT racine pour authentifier le AWS IoT serveur avec lequel ils communiquent. Pour de plus amples informations, veuillez consulter Authentification du serveur.

- Une AWS IoT règle.

  Une règle contient une requête et une ou plusieurs actions de règle. La requête extrait les données des messages de l'appareil pour déterminer si les données du message doivent être traitées. L'action de règle spécifie ce qu'il faut faire si les données correspondent à la requête.

- Une rubrique Amazon SNS et une souscription de rubrique.

  La règle écoute les données d'humidité de votre Raspberry Pi. Si la valeur est inférieure à un seuil, un message est envoyé dans la rubrique Amazon SNS. Amazon SNS envoie ce message à toutes les adresses e-mail abonnées au sujet.

Étape 1 : Création de la AWS IoT politique

Créez une AWS IoT politique qui permet à votre Raspberry Pi de se connecter et d'envoyer des messages à AWS IoT.

1. Dans la console AWS IoT, si un bouton Commencer s'affiche, appuyez dessus. Dans le panneau de navigation du service, développez Sécurité, puis choisissez Politiques.
2. Si une boîte de dialogue Vous ne possédez pas encore de stratégie s'affiche, choisissez Créer une stratégie. Sinon, cliquez sur Create.
3. Entrez le nom de la AWS IoT politique (par exemple,**`MoistureSensorPolicy`**).
4. Dans la section Ajouter des instructions, remplacez la stratégie existante par le code JSON suivant. Remplacez *`region`* et *`account`* par votre Compte AWS numéro Région AWS et.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [{
         "Effect": "Allow",
         "Action": "iot:Connect",
         "Resource": "arn:aws:iot:region:account:client/RaspberryPi"
      },
      {
         "Effect": "Allow",
         "Action": "iot:Publish",
         "Resource": [
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
update",
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
delete",
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
get"
         ]
      },
      {
         "Effect": "Allow",
         "Action": "iot:Receive",
         "Resource": [
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
update/accepted",
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
delete/accepted",
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
get/accepted",
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
update/rejected",
            "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/
delete/rejected"
         ]
      },
      {
         "Effect": "Allow",
         "Action": "iot:Subscribe",
         "Resource": [
```

```
            "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/
shadow/update/accepted",
            "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/
shadow/delete/accepted",
            "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/
shadow/get/accepted",
            "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/
shadow/update/rejected",
            "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/
shadow/delete/rejected"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "iot:GetThingShadow",
            "iot:UpdateThingShadow",
            "iot:DeleteThingShadow"
         ],
         "Resource": "arn:aws:iot:region:account:thing/RaspberryPi"

      }
   ]
}
```

5. Choisissez Créer.

Étape 2 : Création de l' AWS IoT objet, du certificat et de la clé privée

Créez un objet dans le AWS IoT registre pour représenter votre Raspberry Pi.

1. Dans la console AWS IoT, dans le panneau de navigation, choisissez Gérer, puis Objets.
2. Si une boîte de dialogue Vous n'avez pas encore d'objets s'affiche, choisissez Enregistrer un objet. Sinon, cliquez sur Create.
3. Sur la page Création d' AWS IoT objets, choisissez Créer un objet unique.
4. Sur la page Add your device to the device registry (Ajouter votre appareil au registre des appareils), entrez un nom pour votre objet IoT (par exemple, **`RaspberryPi`**), puis choisissez Next (Suivant). Vous ne pouvez pas modifier le nom d'un objet après l'avoir créé. Pour changer le nom d'un objet, vous devez créer un objet, lui donner un nouveau nom, puis supprimer l'ancien objet.

5. Sur la page Add a certificate for your thing (Ajouter un certificat pour votre objet), choisissez Create certificate (Créer un certificat).
6. Choisissez les liens Télécharger pour télécharger le certificat, la clé privée et le certificat CA racine.

   > **Important**
   >
   > C'est la seule fois que vous pouvez télécharger votre certificat et votre clé privée.

7. Choisissez Activer pour activer votre certificat. Le certificat doit être actif pour qu'un appareil puisse se connecter à AWS IoT.
8. Choisissez Attacher une stratégie.
9. Pour Ajouter une politique pour votre objet, choisissez MoistureSensorPolicy, puis choisissez Enregistrer l'objet.

3e étape : Créer une rubrique Amazon SNS et s'abonner

Créer une rubrique Amazon SNS et s'abonner

1. Dans la AWS console SNS, dans le volet de navigation, choisissez Rubriques, puis Créer un rôle.
2. Choisissez le type Standard et entrez un nom pour le sujet (par exemple,**`MoistureSensorTopic`**).
3. Entrez un nom d'affichage pour la rubrique (par exemple, **`Moisture Sensor Topic`**). Il s'agit du nom affiché pour votre rubrique dans la console Amazon SNS.
4. Choisissez Créer une rubrique.
5. Sur la page des détails de la rubrique , sélectionnez Créer un abonnement.
6. Pour Protocole, choisissez E-mail.
7. Saisissez votre adresse e-mail dans Endpoint (Point de terminaison).
8. Choisissez Créer un abonnement.
9. Ouvrez votre client de messagerie et recherchez un message avec l'objet **`MoistureSensorTopic`**. Ouvrez cet e-mail et cliquez sur le lien Confirmer l'abonnement.

> **Important**
>
> Vous ne recevrez aucune alerte par e-mail de cette rubrique Amazon SNS tant que vous n'aurez pas confirmé l'abonnement.

Vous devriez recevoir un message électronique contenant le texte que vous avez saisi.

Étape 4 : créer une AWS IoT règle pour envoyer un e-mail

Une AWS IoT règle définit une requête et une ou plusieurs actions à effectuer lorsqu'un message est reçu d'un appareil. Le moteur de AWS IoT règles écoute les messages envoyés par les appareils et utilise les données contenues dans les messages pour déterminer si des mesures doivent être prises. Pour de plus amples informations, veuillez consulter Règles pour AWS IoT.

Dans ce didacticiel, votre Raspberry Pi publie des messages sur `aws/things/RaspberryPi/shadow/update`. Il s'agit d'une rubrique MQTT interne utilisée par les appareils et le service Thing Shadow. Le Raspberry Pi publie des messages sous la forme suivante :

```
{
    "reported": {
        "moisture" : moisture-reading,
        "temp" : temperature-reading
    }
}
```

Vous créez une requête qui extrait les données d'humidité et de température du message entrant. Vous créez également une action Amazon SNS qui prend les données et les envoie aux abonnés de la rubrique Amazon SNS si le relevé d'humidité est inférieur à un seuil.

Créer une rubrique Amazon SNS

1. Dans la AWS IoT console, choisissez Routage des messages, puis Règles. Si une boîte de dialogue Vous ne possédez pas encore de règle s'affiche, choisissez Créer une règle. Sinon, choisissez Créer une règle.
2. Dans la page des propriétés de la règle, entrez un Nom de règle tel que**`MoistureSensorRule`**, et fournissez une brève description de la règle, telle que**`Sends an alert when soil moisture level readings are too low`**.

3. Choisissez Next et configurez votre instruction SQL. Choisissez la version SQL 2016-03-23 et entrez l'instruction de requête AWS IoT SQL suivante :

   ```
   SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
    state.reported.moisture < 400
   ```

   Cette instruction déclenche l'action de la règle lorsque la valeur de `moisture` est inférieure à `400`.

   > **Note**
   >
   > Vous devrez peut-être utiliser une valeur différente. Une fois que vous avez le code qui s'exécute sur votre Raspberry Pi, vous pouvez voir les valeurs que vous obtenez de votre capteur en touchant le capteur, en le plaçant dans l'eau ou en le plaçant dans un pot.

4. Choisissez Next et associez des actions de règle. Pour l'action 1, choisissez Simple Notification Service. La description de cette action de règle est Envoyer un message sous forme de notification push SNS.
5. Pour le sujet SNS, choisissez le sujet dans 3e étape : Créer une rubrique Amazon SNS et s'abonner lequel vous avez créé le message et laissez le format du message au format RAW. MoistureSensorTopic Pour Rôle IAM, choisissez Créer un rôle. Entrez un nom pour le rôle par exemple, **`LowMoistureTopicRole`**, puis choisissez Créer un rôle.
6. Cliquez sur Suivant pour passer en revue, puis sur Créer pour créer la règle.

Configuration de votre Raspberry Pi et du capteur d'humidité

Insérez votre carte micro SD dans le Raspberry Pi, connectez votre écran, votre clavier, votre souris et, si vous n'utilisez pas le Wi-Fi, un câble Ethernet. Ne connectez pas encore le câble d'alimentation.

Connectez le câble jumper JST au capteur d'humidité. L'autre côté du jumper a quatre câbles :

- Vert : I2C SCL
- Blanc : I2C SDA
- Rouge : alimentation (3,5 V)

- Black : terre

Maintenez la carte Raspberry Pi enfoncée avec la prise Ethernet sur la droite. Dans cette orientation, il y a deux lignes de broches GPIO en haut. Connectez les câbles du capteur d'humidité à la ligne inférieure de broches dans l'ordre suivant. À partir du connecteur le plus à gauche, connectez rouge (alimentation), blanc (SDA) et vert (SCL). Ignorez une broche, puis connectez le fil noir (terre). Pour plus d'informations, consultez Câblage informatique Python.

Attachez le câble d'alimentation au Raspberry Pi et branchez l'autre extrémité à une prise murale pour l'allumer.

Configuration de votre Raspberry Pi

1. Sur Welcome to Raspberry Pi (Bienvenue dans Raspberry Pi), choisissez Next (Suivant).
2. Choisissez votre pays, votre langue, votre fuseau horaire et votre disposition du clavier. Choisissez Suivant.
3. Saisissez un mot de passe pour votre Raspberry Pi, puis choisissez Next (Suivant).
4. Choisissez votre réseau Wi-Fi, puis choisissez Next (Suivant). Si vous n'utilisez pas de réseau Wi-Fi, choisissez Skip (Ignorer).
5. Choisissez Next (Suivant) pour rechercher les mises à jour logicielles. Lorsque les mises à jour sont terminées, choisissez Restart (Redémarrer) pour redémarrer votre Raspberry Pi.

Une fois que votre Raspberry Pi a démarré, activez l'interface I2C.

1. Dans le coin supérieur gauche du bureau Raspbian, cliquez sur l'icône Raspberry, choisissez Preferences (Préférences), puis Raspberry Pi Configuration (Configuration du Raspberry Pi).
2. Sous l'onglet Interfaces pour I2C, choisissez Enable (Activer).
3. Choisissez OK.

Les bibliothèques du capteur d'humidité Adafruit STEMMA ont été écrites pour. CircuitPython Pour les exécuter sur un Raspberry Pi, vous devez installer la dernière version de Python 3.

1. Exécutez les commandes suivantes à partir d'une invite de commande pour mettre à jour votre logiciel Raspberry Pi :

   ```
   sudo apt-get update
   ```

```
sudo apt-get upgrade
```

2. Exécutez la commande suivante pour mettre à jour votre installation Python 3 :

   ```
   sudo pip3 install --upgrade setuptools
   ```

3. Exécutez la commande suivante pour installer les bibliothèques GPIO Raspberry Pi :

   ```
   pip3 install RPI.GPIO
   ```

4. Exécutez la commande suivante pour installer les bibliothèques Adafruit Blinka :

   ```
   pip3 install adafruit-blinka
   ```

   Pour plus d'informations, consultez Installation de CircuitPython bibliothèques sur le Raspberry Pi.

5. Exécutez la commande suivante pour installer les bibliothèques Adafruit Seesaw :

   ```
   sudo pip3 install adafruit-circuitpython-seesaw
   ```

6. Exécutez la commande suivante pour installer le AWS IoT Device SDK pour Python :

   ```
   pip3 install AWSIoTPythonSDK
   ```

Votre Raspberry Pi dispose désormais de toutes les bibliothèques requises. Créez un fichier appelé **`moistureSensor.py`** et copiez le code Python suivant dans le fichier :

```
from adafruit_seesaw.seesaw import Seesaw
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
from board import SCL, SDA

import logging
import time
import json
import argparse
import busio

# Shadow JSON schema:
#
# {
#   "state": {
#       "desired":{
```

```
#           "moisture":<INT VALUE>,
#           "temp":<INT VALUE>
#       }
#   }
# }

# Function called when a shadow is updated
def customShadowCallback_Update(payload, responseStatus, token):

    # Display status and data from update request
    if responseStatus == "timeout":
        print("Update request " + token + " time out!")

    if responseStatus == "accepted":
        payloadDict = json.loads(payload)
        print("~~~~~~~~~~~~~~~~~~~~~~~")
        print("Update request with token: " + token + " accepted!")
        print("moisture: " + str(payloadDict["state"]["reported"]["moisture"]))
        print("temperature: " + str(payloadDict["state"]["reported"]["temp"]))
        print("~~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Update request " + token + " rejected!")

# Function called when a shadow is deleted
def customShadowCallback_Delete(payload, responseStatus, token):

     # Display status and data from delete request
    if responseStatus == "timeout":
        print("Delete request " + token + " time out!")

    if responseStatus == "accepted":
        print("~~~~~~~~~~~~~~~~~~~~~~~")
        print("Delete request with token: " + token + " accepted!")
        print("~~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Delete request " + token + " rejected!")


# Read in command-line parameters
def parseArgs():

    parser = argparse.ArgumentParser()
```

```
    parser.add_argument("-e", "--endpoint", action="store", required=True, dest="host",
 help="Your device data endpoint")
    parser.add_argument("-r", "--rootCA", action="store", required=True,
 dest="rootCAPath", help="Root CA file path")
    parser.add_argument("-c", "--cert", action="store", dest="certificatePath",
 help="Certificate file path")
    parser.add_argument("-k", "--key", action="store", dest="privateKeyPath",
 help="Private key file path")
    parser.add_argument("-p", "--port", action="store", dest="port", type=int,
 help="Port number override")
    parser.add_argument("-n", "--thingName", action="store", dest="thingName",
 default="Bot", help="Targeted thing name")
    parser.add_argument("-id", "--clientId", action="store", dest="clientId",
 default="basicShadowUpdater", help="Targeted client id")

    args = parser.parse_args()
    return args


# Configure logging
# AWSIoTMQTTShadowClient writes data to the log
def configureLogging():

    logger = logging.getLogger("AWSIoTPythonSDK.core")
    logger.setLevel(logging.DEBUG)
    streamHandler = logging.StreamHandler()
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
 %(message)s')
    streamHandler.setFormatter(formatter)
    logger.addHandler(streamHandler)


# Parse command line arguments
args = parseArgs()

if not args.certificatePath or not args.privateKeyPath:
    parser.error("Missing credentials for authentication.")
    exit(2)

# If no --port argument is passed, default to 8883
if not args.port:
    args.port = 8883
```

```
# Init AWSIoTMQTTShadowClient
myAWSIoTMQTTShadowClient = None
myAWSIoTMQTTShadowClient = AWSIoTMQTTShadowClient(args.clientId)
myAWSIoTMQTTShadowClient.configureEndpoint(args.host, args.port)
myAWSIoTMQTTShadowClient.configureCredentials(args.rootCAPath, args.privateKeyPath,
 args.certificatePath)

# AWSIoTMQTTShadowClient connection configuration
myAWSIoTMQTTShadowClient.configureAutoReconnectBackoffTime(1, 32, 20)
myAWSIoTMQTTShadowClient.configureConnectDisconnectTimeout(10) # 10 sec
myAWSIoTMQTTShadowClient.configureMQTTOperationTimeout(5) # 5 sec

# Initialize Raspberry Pi's I2C interface
i2c_bus = busio.I2C(SCL, SDA)

# Intialize SeeSaw, Adafruit's Circuit Python library
ss = Seesaw(i2c_bus, addr=0x36)

# Connect to AWS IoT
myAWSIoTMQTTShadowClient.connect()

# Create a device shadow handler, use this to update and delete shadow document
deviceShadowHandler =
 myAWSIoTMQTTShadowClient.createShadowHandlerWithName(args.thingName, True)

# Delete current shadow JSON doc
deviceShadowHandler.shadowDelete(customShadowCallback_Delete, 5)

# Read data from moisture sensor and update shadow
while True:

    # read moisture level through capacitive touch pad
    moistureLevel = ss.moisture_read()

    # read temperature from the temperature sensor
    temp = ss.get_temp()

    # Display moisture and temp readings
    print("Moisture Level: {}".format(moistureLevel))
    print("Temperature: {}".format(temp))

    # Create message payload
    payload = {"state":{"reported":{"moisture":str(moistureLevel),"temp":str(temp)}}}
```

```
    # Update shadow
    deviceShadowHandler.shadowUpdate(json.dumps(payload), customShadowCallback_Update,
5)
    time.sleep(1)
```

Enregistrez le fichier dans un emplacement où vous le trouverez. Sur la ligne de commande, tapez `moistureSensor.py` avec les paramètres suivants :

point de terminaison

Votre point de AWS IoT terminaison personnalisé. Pour de plus amples informations, veuillez consulter API REST Device Shadow.

rootCA

Le chemin complet vers votre certificat d'autorité de certification AWS IoT racine.

cert

Le chemin complet vers le certificat de votre AWS IoT appareil.

clé

Le chemin complet vers la clé privée du certificat de votre AWS IoT appareil.

thingName

Votre nom d'objet (dans ce cas, `RaspberryPi`).

clientId

ID du client MQTT. Utilisez `RaspberryPi`.

La ligne de commande doit se présenter comme suit :

```
python3 moistureSensor.py --endpoint your-endpoint --rootCA ~/certs/
AmazonRootCA1.pem --cert ~/certs/raspberrypi-certificate.pem.crt --key
~/certs/raspberrypi-private.pem.key --thingName RaspberryPi --clientId
RaspberryPi
```

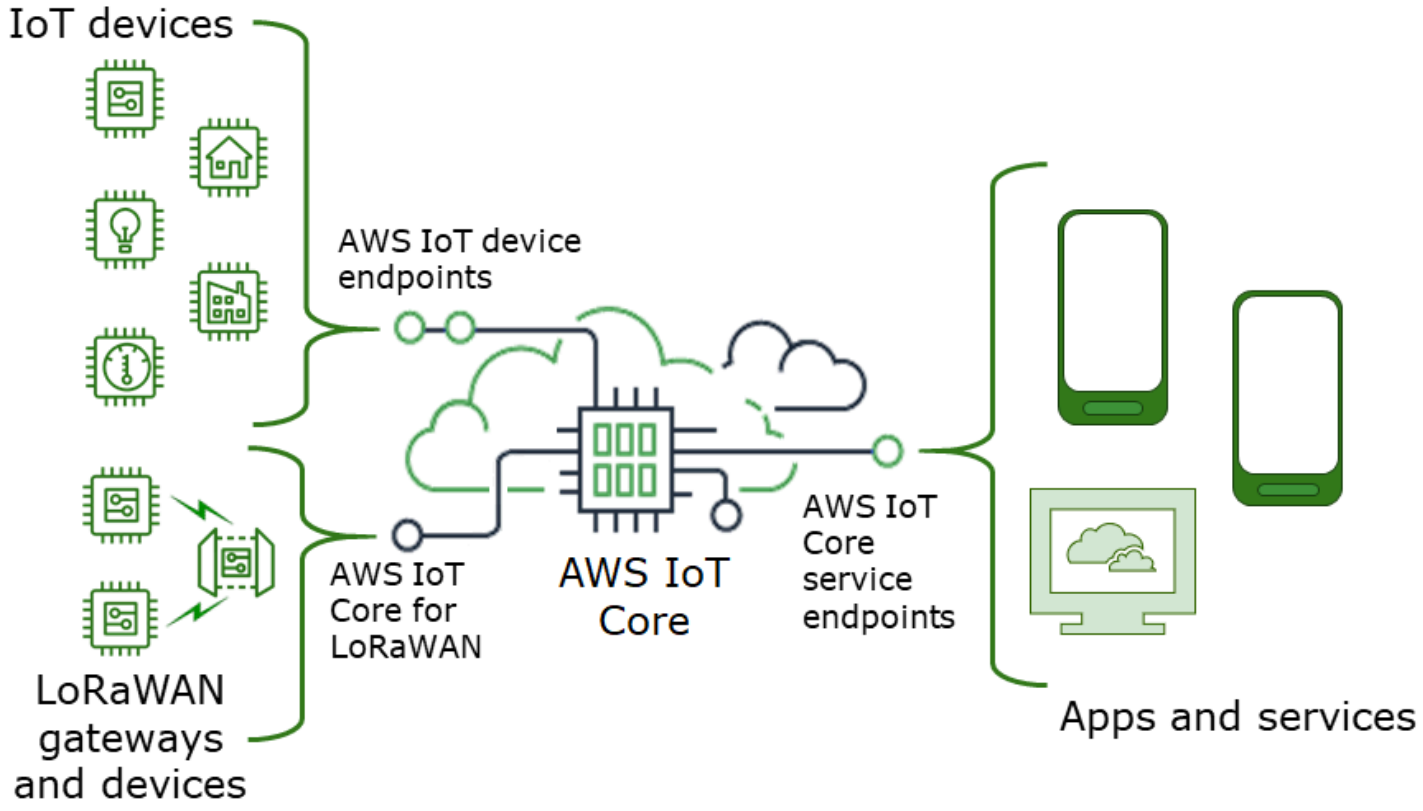
Essayez de toucher le capteur, de le placer dans un pot ou de le placer dans un verre d'eau pour voir comment le capteur réagit à différents niveaux d'humidité. Si nécessaire, vous pouvez modifier la valeur de seuil dans `MoistureSensorRule`. Lorsque la valeur du capteur d'humidité est inférieure à la valeur spécifiée dans l'instruction de requête SQL de votre règle, AWS IoT un message est publié

dans la rubrique Amazon SNS. Vous devez recevoir un e-mail contenant les données relatives à l'humidité et à la température.

Une fois que vous avez vérifié la réception des e-mails provenant de , appuyez sur CTRL+C pour arrêter le programme Python. Il est peu probable que le programme Python envoie suffisamment de messages pour générer des frais, mais il est recommandé d'arrêter le programme une fois que vous avez terminé.

Connect à AWS IoT Core

AWS IoT Core prend en charge les connexions avec les appareils IoT, les passerelles sans fil, les services et les applications. Les appareils se connectent pour AWS IoT Core pouvoir envoyer des données à des AWS IoT services et à d'autres appareils et en recevoir en provenance de ceux-ci. Les applications et autres services se connectent également AWS IoT Core pour contrôler et gérer les appareils IoT et traiter les données de votre solution IoT. Cette section explique comment choisir le meilleur moyen de connexion et de communication AWS IoT Core pour chaque aspect de votre solution IoT.



Il existe plusieurs manières d'interagir avec AWS IoT. Les applications et les services peuvent utiliser AWS IoT Core- extrémités du point de terminaison du plan de contrôle et les appareils peuvent se connecter AWS IoT Core en utilisant AWS IoT points de terminaison de l'appareil ou AWS IoT Core pour les régions LoRa WAN et les points de terminaison.

AWS IoT Core- extrémités du point de terminaison du plan de contrôle

Les AWS IoT Core points de terminaison du plan de contrôle permettent d'accéder aux fonctions qui contrôlent et gèrent votre AWS IoT solution.

- Points de terminaison

  Les AWS IoT Core - points de terminaison du AWS IoT Core plan de contrôle et du plan de contrôle Device Advisor sont spécifiques à la région et sont répertoriés dans AWS IoT Core Points de terminaison et quotas. Les formats des points de terminaison sont les suivants.

Objectif du point de terminaison	Format du point de terminaison	Sert
AWS IoT Core - Plan de contrôle	`iot.aws-region.amazonaws.com`	AWS IoT API Control Plane
AWS IoT Core Device Advisor - plan de contrôle	`api.iotdeviceadvisor.aws-region.amazonaws.com`	AWS IoT Core API Device Advisor Control Plane

- SDKs et outils

  Ils AWS SDKsfournissent un support spécifique à la langue pour le AWS IoT Core APIs et pour les APIs autres AWS services. The AWS Mobile SDKs fournit aux développeurs d'applications une assistance spécifique à la plate-forme pour l' AWS IoT Core API et d'autres AWS services sur les appareils mobiles.

  AWS CLIFournit un accès en ligne de commande aux fonctions fournies par les points de terminaison du AWS IoT service. AWS Tools for PowerShell fournit des outils pour gérer les AWS services et les ressources dans l'environnement PowerShell de script.

- Authentification

  Les points de terminaison du service utilisent les utilisateurs et les AWS informations d'identification IAM pour authentifier les utilisateurs.

- En savoir plus

Pour plus d'informations et les liens vers les références du SDK, consultez the section called "Connectez-vous aux points de terminaison AWS IoT Core du service".

AWS IoT points de terminaison de l'appareil

Les points de terminaison de l' AWS IoT appareil prennent en charge la communication entre vos appareils IoT et AWS IoT.

- Points de terminaison

  Le support AWS IoT Core et les AWS IoT Device Management fonctions des terminaux de l'appareil. Ils sont spécifiques à vous Compte AWS et vous pouvez les voir en utilisant la describe-endpoint commande.

Objectif du point de terminaison	Format du point de terminaison	Sert
AWS IoT Core - plan de données	Consultez ???.	AWS IoT API Data Plane
AWS IoT Device Management- données sur les tâches	Consultez ???.	AWS IoT API Jobs Data Plane
AWS IoT Device Advisor - plan de données	Consultez ???.	Ne s'applique pas
AWS IoT Device Management - Fleet Hub	Ne s'applique pas	Ne s'applique pas
AWS IoT Device Management - Tunneling sécurisé	`api.tunneling.iot.` *`aws-region`* `.amazonaws.com`	AWS IoT API de tunneling sécurisé

  Pour plus d'informations sur ces points de terminaison et les fonctions qu'ils prennent en charge, consultezthe section called "AWS IoT données de l'appareil et points de terminaison de service".

- SDKs

L'AWS IoT appareil SDKs fournit un support spécifique à la langue pour les protocoles MQTT (Message Queueing Telemetry Transport) et WebSocket Secure (WSS), avec lesquels les appareils communiquent. AWS IoTAWS Applications mobiles SDKsfournissent également un support pour les communications sur les appareils MQTT et pour APIs d'autres AWS services sur les appareils mobiles. AWS IoT APIs

- Authentification

  Les points de terminaison de l'appareil utilisent des certificats X.509 ou des utilisateurs AWS IAM dotés d'informations d'identification pour authentifier les utilisateurs.

- En savoir plus

  Pour plus d'informations et les liens vers les références du SDK, consultez the section called "AWS IoT Device SDKs".

AWS IoT Core pour les passerelles et appareils LoRa WAN

AWS IoT Core pour le LoRa WAN connecte des passerelles et des appareils sans fil à AWS IoT Core.

- Points de terminaison

  AWS IoT Core for LoRa WAN gère les connexions de passerelle vers les points de terminaison spécifiques au compte et à la région AWS IoT Core . Les passerelles peuvent se connecter au point de terminaison du serveur de configuration et de mise à jour (CUPS) de votre compte fourni AWS IoT Core par le LoRa WAN.

Objectif du point de terminaison	Format du point de terminaison	Sert
Serveur de configuration et de mise à jour (CUPS)	*account-specific-prefix* `.cups.lorawan.` *aws-region* `.amazonaws.com:443`	Communication par passerelle avec le serveur de configuration et de mise à jour fourni par AWS IoT Core for LoRa WAN
LoRaServeur réseau WAN (LNS)	*account-specific-prefix* `.gateway.`	Communication par passerelle avec le serveur réseau

Objectif du point de terminaison	Format du point de terminaison	Sert
	`lorawan.` *`aws-region`* `.amazonaws.com:443`	LoRa WAN fournie par AWS IoT Core for LoRa WAN

- SDKs

  L'API AWS IoT sans fil sur laquelle AWS IoT Core repose le LoRa WAN est prise en charge par le AWS SDK. Pour plus d'informations, voir AWS SDKs et Boîtes à outils.

- Authentification

  AWS IoT Core pour les communications entre appareils LoRa WAN, utilisez des certificats X.509 pour sécuriser les communications avec AWS IoT.

- En savoir plus

  Pour plus d'informations sur la configuration et la connexion de périphériques sans fil, reportez-vous AWS IoT Core à la section Régions LoRa WAN et points de terminaison.

Connectez-vous aux points de terminaison AWS IoT Core du service

Vous pouvez accéder aux fonctionnalités du AWS IoT Core plan de contrôle en utilisant le AWS CLI AWS SDK correspondant à votre langue préférée ou en appelant directement l'API REST. Nous vous recommandons d'utiliser le AWS CLI ou un AWS SDK pour interagir, AWS IoT Core car ils intègrent les meilleures pratiques en matière de AWS services d'appel. Il est possible d'appeler APIs directement le REST, mais vous devez fournir les informations de sécurité nécessaires pour accéder à l'API.

> **Note**
>
> Les appareils IoT devraient utiliser AWS IoT Device SDKs. SDKs Les appareils sont optimisés pour une utilisation sur les appareils, prennent en charge la communication MQTT et prennent en charge les appareils les AWS IoT APIs plus utilisés par les appareils. AWS IoT Pour plus d'informations sur l'appareil SDKs et les fonctionnalités qu'il fournit, consultezAWS IoT Device SDKs.

Les appareils mobiles devraient utiliser AWS Applications mobiles SDKs. The Mobile SDKs fournit un support pour AWS IoT APIs les communications avec les appareils MQTT et pour APIs d'autres AWS services sur les appareils mobiles. Pour plus d'informations sur le mobile SDKs et les fonctionnalités qu'il fournit, consultezAWS Applications mobiles SDKs.

Vous pouvez utiliser AWS Amplify les outils et les ressources des applications Web et mobiles pour vous connecter plus facilement à AWS IoT Core. Pour plus d'informations sur la connexion à AWS IoT Core Amplify à l'aide d'Amplify, consultez la PubSubdocumentation Amplify.

Les sections suivantes décrivent les outils SDKs que vous pouvez utiliser pour développer et interagir avec AWS IoT d'autres AWS services. Pour obtenir la liste complète des AWS outils et kits de développement disponibles pour créer et gérer des applications AWS, consultez la section Outils sur lesquels vous pouvez vous appuyer AWS.

AWS CLI pour AWS IoT Core

AWS CLI Fournit un accès en ligne de commande à. AWS APIs

- Installation

  Pour plus d'informations sur l'installation du AWS CLI, voir Installation du AWS CLI.
- Authentification

  Il AWS CLI utilise les informations d'identification de votre Compte AWS.
- Référence

  Pour plus d'informations sur les AWS CLI commandes de ces AWS IoT Core services, voir :
  - AWS CLI Référence de commande pour l'IoT
  - AWS CLI Référence de commande pour les données IoT
  - AWS CLI Command Reference pour les données relatives aux emplois liés à l'IoT
  - AWS CLI Référence de commande pour le tunneling sécurisé de l'IoT

Pour les outils permettant de gérer les AWS services et les ressources dans l'environnement PowerShell de script, voir AWS Outils pour PowerShell.

AWS SDKs

Avec AWS SDKs, vos applications et appareils compatibles peuvent appeler AWS IoT APIs et accéder à APIs d'autres AWS services. Cette section fournit des liens vers AWS SDKs et vers la documentation de référence APIs de l'API pour les AWS IoT Core services.

Ils AWS SDKs les soutiennent AWS IoT Core APIs

- AWS IoT
- AWS IoT Plan de données
- AWS IoT Jobs Data Plane
- AWS IoT Tunneling sécurisé
- AWS IoT Sans fil

C++

Pour installer le AWS SDK pour C++ et l'utiliser pour vous connecter à AWS IoT :

1. Suivez les instructions de la section Getting Started Using the AWS SDK for C++

   Ces instructions décrivent comment :

   - Installer et générer du SDK à partir de fichiers sources
   - Fournir les informations d'identification pour utiliser le SDK avec votre Compte AWS
   - Initialiser et arrêter le SDK dans votre application ou votre service
   - Créez un CMake projet pour créer votre application ou votre service
2. Afficher et exécuter un exemple de requête. Pour des exemples d'applications utilisant le SDK AWS pour C++, consultez la section AWS SDK pour C++ Exemples de code.

Documentation pour les AWS IoT Core services pris en AWS SDK pour C++ charge

- AWS documentation de référence TClient « : :Io »
- Documentation de référence Aws : :Io TData Plane : :Io TData PlaneClient
- Documentation de référence Aws : :Io TJobs DataPlane : :Io TJobs DataPlaneClient
- Documentation de référence Aws : :Io TSecure Tunneling : :Io TSecure TunnelingClient

Go

Pour installer le AWS SDK pour Go et l'utiliser pour vous connecter à AWS IoT :

1. Suivez les instructions de la section Getting Started with AWS SDK pour Go

   Ces instructions décrivent comment :

   - Installez le AWS SDK pour Go
   - Obtenez des clés d'accès pour que le SDK accède à votre Compte AWS
   - Importez des packages dans le code source de nos applications ou services
2. Afficher et exécuter un exemple de requête. Pour des exemples d'applications utilisant le AWS SDK pour Go, consultez la section AWS SDK pour Go Exemples de code.

Documentation pour les AWS IoT Core services pris en AWS SDK pour Go charge

- Documentation de référence du kit IoT
- Documentation de référence Io TData Plane
- Documentation de TJobs DataPlane référence Io
- Documentation de référence sur Io TSecure Tunneling

Java

Pour installer le AWS SDK pour Java et l'utiliser pour vous connecter à AWS IoT :

1. Suivez les instructions de la section Getting Started with AWS SDK for Java 2.x

   Ces instructions décrivent comment :

   - Inscrivez-vous AWS et créez un utilisateur IAM
   - Téléchargement du kit SDK
   - Configurer les AWS informations d'identification et la région
   - Utilisation du kit SDK avec Apache Maven
   - Utilisation du kit SDK avec Gradle
2. Créez et exécutez un exemple d'application à l'aide de l'un des AWS SDK for Java 2.x Exemples de Codes.
3. Consultez la documentation de référence de l'API SDK

Documentation pour les AWS IoT Core services pris en AWS SDK pour Java charge

- IotClient documentation de référence
- IotDataPlaneClient documentation de référence
- IotJobsDataPlaneClient documentation de référence
- Documentation de TSecure TunnelingClient référence Io

JavaScript

Pour installer le AWS SDK pour JavaScript et l'utiliser pour vous connecter à AWS IoT :

1. Suivez les instructions de pour Configurer le AWS SDK pour JavaScript. Ces instructions s'appliquent à l'utilisation du AWS SDK pour JavaScript dans le navigateur et avec Node.JS. Assurez-vous de suivre les instructions qui s'appliquent à votre installation.

   Ces instructions décrivent comment :

   - Remplir les conditions préalables
   - Installez le SDK pour JavaScript
   - Chargez le SDK pour JavaScript
2. Créez et exécutez un exemple d'application pour démarrer avec le SDK, comme le décrit l'option de démarrage pour votre environnement.
   - Commencez avec le AWS SDK pour JavaScript in the Browser, ou
   - Commencez avec le AWS SDK pour JavaScript dans Node.js

Documentation pour les AWS IoT Core services pris en AWS SDK pour JavaScript charge

- `AWS.Iot reference documentation`
- `AWS.IotData reference documentation`
- `AWS.IotJobsDataPlane reference documentation`
- `AWS.IotSecureTunneling reference documentation`

.NET

Pour installer le AWS SDK pour .NET et l'utiliser pour vous connecter à AWS IoT :

1. Suivez les instructions de la section Configuration de votre AWS SDK pour .NET environnement
2. Suivez les instructions de la section Configuration de votre AWS SDK pour .NET projet

   Ces instructions décrivent comment :

   - Lancement d'un nouveau projet
   - Obtenir et configurer les AWS informations d'identification
   - Installer les packages du AWS SDK
3. Créez et exécutez l'un des exemples de programmes de la section Travailler avec les AWS services dans le AWS SDK for .NET
4. Consultez la documentation de référence de l'API SDK

Documentation pour les AWS IoT Core services pris en AWS SDK pour .NET charge

- Documentation de référence Amazon.IOT.Model
- Amazon. IotData.Documentation de référence du modèle
- Documentation de référence Amazon.IO TJobs DataPlane .Model
- Documentation de référence du modèle Amazon.IO TSecure Tunneling.Model

PHP

Pour installer le AWS SDK pour PHP et l'utiliser pour vous connecter à AWS IoT :

1. Suivez les instructions de la section Mise en route avec la AWS SDK pour PHP version 3

   Ces instructions décrivent comment :

   - Remplir les conditions préalables
   - Installer le SDK
   - Appliquer le SDK à un script PHP
2. Créez et exécutez un exemple d'application à l'aide de l'un des AWS SDK pour PHP Exemples de codes version 3

Documentation pour les AWS IoT Core services pris en AWS SDK pour PHP charge

- Documentation de TClient référence Io
- Documentation de TData PlaneClient référence Io
- Documentation de TJobs DataPlaneClient référence Io
- Documentation de TSecure TunnelingClient référence Io

Python

Pour installer le AWS SDK pour Python (Boto3) et l'utiliser pour vous connecter à AWS IoT :

1. Suivez les instructions du AWS SDK pour Python (Boto3) Quickstart

   Ces instructions décrivent comment :

   - Installer le SDK
   - Configuration du kit SDK
   - Utilisez le SDK dans votre code
2. Créez et exécutez un exemple de programme qui utilise le AWS SDK pour Python (Boto3)

   Ce programme affiche les options de journalisation actuellement configurées pour le compte. Après avoir installé le SDK et l'avoir configuré pour votre compte, vous devriez être en mesure d'exécuter ce programme.

   ```
   import boto3
   import json

   # initialize client
   iot = boto3.client('iot')

   # get current logging levels, format them as JSON, and write them to stdout
   response = iot.get_v2_logging_options()
   print(json.dumps(response, indent=4))
   ```

   Pour obtenir plus d'informations sur les fonctions utilisées dans cet exemple, consultez the section called “Configuration de la AWS IoT journalisation”.

Documentation pour les AWS IoT Core services pris en AWS SDK pour Python (Boto3) charge

- Documentation de référence du kit IoT
- Documentation de référence Io TData Plane
- Documentation de TJobs DataPlane référence Io
- Documentation de référence sur Io TSecure Tunneling

Ruby

Pour installer le AWS SDK pour Ruby et l'utiliser pour vous connecter à AWS IoT :

- Suivez les instructions de la section Getting Started with AWS SDK pour Ruby

  Ces instructions décrivent comment :
  - Installer le SDK
  - Configuration du kit SDK
- Créez et exécutez le didacticiel Hello World

Documentation des AWS IoT Core services pris en charge par le AWS SDK for Ruby

- Documentation de référence Aws : :IoT : :Client
- Documentation de référence Aws : :Io TData Plane : :Client
- Documentation de référence Aws : :Io TJobs DataPlane : :Client
- Documentation de référence Aws : :Io TSecure Tunneling : :Client

AWS Applications mobiles SDKs

The AWS Mobile SDKs fournit aux développeurs d'applications mobiles une assistance spécifique à la plate-forme pour les services AWS IoT Core , APIs la communication entre appareils IoT à l'aide de MQTT et les APIs autres services. AWS

Android

AWS Mobile SDK for Android

AWS Mobile SDK for Android Il contient une bibliothèque, des exemples et de la documentation permettant aux développeurs de créer des applications mobiles connectées à l'aide de AWS. Ce

SDK inclut également la prise en charge des communications entre appareils MQTT et APIs de l' AWS IoT Core appel des services. Pour plus d'informations, consultez les ressources suivantes :

- AWS SDK mobile pour Android sur GitHub
- AWS SDK mobile pour Android Readme
- AWS Exemples de SDK mobile pour Android
- AWS Référence de l'API SDK for Android
- AWSIoTClient Documentation de référence sur les classes

iOS

AWS Mobile SDK for iOS

AWS Mobile SDK for iOS Il s'agit d'un kit de développement logiciel open source, distribué sous licence Apache Open Source. Le SDK pour iOS fournit une bibliothèque, des exemples de code et de la documentation pour aider les développeurs à créer des applications AWS mobiles connectées à l'aide de. Ce SDK inclut également la prise en charge des communications entre appareils MQTT et APIs de l' AWS IoT Core appel des services. Pour plus d'informations, consultez les ressources suivantes :

- AWS Mobile SDK for iOS sur GitHub
- AWS SDK pour iOS Readme
- AWS Exemples de SDK pour iOS
- AWS IoT Documents de référence des classes dans le AWS SDK pour iOS

RESTE APIs des AWS IoT Core services

Le REST APIs des AWS IoT Core services peut être appelé directement à l'aide de requêtes HTTP.

- URL de point de terminaison

  Les points de terminaison de service qui exposent le reste APIs des AWS IoT Core services varient selon les régions et sont répertoriés dans AWS IoT Core Points de terminaison et quotas. Vous devez utiliser le point de terminaison de la région qui possède les AWS IoT ressources auxquelles vous souhaitez accéder, car les AWS IoT ressources sont spécifiques à la région.
- Authentification

Le reste APIs des AWS IoT Core services utilise les informations d'identification AWS IAM pour l'authentification. Pour plus d'informations, consultez la section Signature des demandes AWS d'API dans le manuel de référence AWS général.

- Référence d'API

  Pour plus d'informations sur les fonctions spécifiques fournies par le REST APIs des AWS IoT Core services, voir :

  - Référence API pour l'IoT.
  - Référence API pour les données IoT.
  - Référence API pour les données de tâches IoT.
  - Référence API pour le tunneling sécurisé IoT.

Connectez des appareils à AWS IoT

Les appareils se AWS IoT connectent à d'autres services via AWS IoT Core. Les appareils envoient et reçoivent des messages via des points de terminaison spécifiques à votre compte. AWS IoT Core Les communications the section called "AWS IoT Device SDKs" entre appareils de support à l'aide des protocoles MQTT et WSS. Pour plus d'informations sur les protocoles que les appareils peuvent utiliser, consultez the section called "Protocoles de communication des appareils".

L'agent de messages

AWS IoT gère les communications entre les appareils par le biais d'un courtier de messages. Les appareils et les clients publient des messages sur le agent de messages et s'abonnent également aux messages publiés par celui-ci. Les messages sont identifiés par une rubriquedéfinie par l'application. Lorsque l'agent de messages reçoit un message publié par un appareil ou un client, il republie ce message sur les appareils et avec les clients abonnés à la rubrique. Le courtier de messages transmet également les messages au moteur de AWS IoT règles, qui peut agir sur le contenu du message.

AWS IoT sécurité des messages

Connexions aux appareils à AWS IoT utiliser the section called "Certificats client X.509" et AWS signature V4 pour l'authentification. Les communications entre appareils sont sécurisées par la version 1.3 du protocole TLS et AWS IoT nécessitent que les appareils envoient l'extension SNI (Server Name Indication) lorsqu'ils se connectent. Pour plus d'informations, voir Sécurité du transport dans AWS IoT.

AWS IoT données de l'appareil et points de terminaison de service

> **Important**
>
> Vous pouvez mettre en cache ou stocker les points de terminaison sur votre appareil. Cela signifie que vous n'aurez pas besoin de demander l'API `DescribeEndpoint` chaque fois qu'un nouvel appareil est connecté. Les points de terminaison ne changeront pas AWS IoT Core une fois qu'ils auront été créés pour votre compte.

Chaque compte possède plusieurs points de terminaison qui lui sont propres et prennent en charge des fonctions IoT spécifiques. Les points de terminaison de données des AWS IoT appareils prennent en charge un publish/subscribe protocole conçu pour les besoins de communication des appareils IoT ; toutefois, d'autres clients, tels que les applications et les services, peuvent également utiliser cette interface si leur application nécessite les fonctionnalités spécialisées fournies par ces points de terminaison. Les points de terminaison du service des AWS IoT appareils permettent un accès centré sur les appareils aux services de sécurité et de gestion.

Pour connaître le point de terminaison des données de l'appareil de votre compte, vous pouvez le trouver sur la page Paramètres de votre AWS IoT Core console.

Pour connaître le point de terminaison de l'appareil de votre compte dans un but spécifique, y compris le point de terminaison des données de l'appareil, utilisez la commande describe-endpoint CLI illustrée ici ou l'API REST `DescribeEndpoint`, et fournissez la valeur du paramètre *`endpointType`* indiquée dans le tableau suivant.

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

Cette commande renvoie un *`iot-endpoint`* au format suivant :*`account-specific-prefix`*`.iot.`*`aws-region`*`.amazonaws.com`.

Chaque client a `iot:Data-ATS` et un autre type de point de terminaison `iot:Data`. Chaque point de terminaison utilise un certificat X.509 pour authentifier le client. Nous recommandons vivement aux clients d'utiliser le type de point de terminaison `iot:Data-ATS` le plus récent pour éviter les problèmes liés à la méfiance généralisée à l'égard des autorités de certification Symantec. Nous fournissons le `iot:Data` point de terminaison permettant aux appareils de récupérer les données des anciens terminaux qui utilisent des VeriSign certificats à des fins de rétrocompatibilité. Pour de plus amples informations, veuillez consulter Authentification du serveur.

AWS IoT points de terminaison pour appareils

Objectif du point de terminaison	Valeur ***endpointType***	Description
AWS IoT Core - opérations de plan de données	`iot:Data-ATS`	Permet d'envoyer des données vers et de recevoir des données depuis les composants Agent de messages, Device shadow et Moteur de règles de AWS IoT. `iot:Data-ATS` renvoie un point de terminaison de données signées ATS.
AWS IoT Core - opérations du plan de données (anciennes)	`iot:Data`	`iot:Data`renvoie un point de terminaison de données VeriSign signé fourni à des fins de rétrocompatibilité . MQTT 5 n'est pas pris en charge sur les points de terminaison Symantec (`iot:Data`).
AWS IoT Core accès aux informations d'identification	`iot:CredentialProvider`	Utilisé pour échanger le certificat X.509 intégré d'un appareil contre des informations d'identification temporaires permettant de se connecter directement à d'autres services AWS . Pour plus d'informations sur la connexion à d'autres AWS services, voir Autoriser les appels directs vers les AWS services.

Objectif du point de terminaison	Valeur *endpointType*	Description
AWS IoT Device Management- opérations de données sur les tâches	`iot:Jobs`	Utilisé pour permettre aux appareils d'interagir avec le service AWS IoT Jobs à l'aide du Jobs Device HTTPS APIs.
AWS IoT Opérations de Device Advisor	`iot:DeviceAdvisor`	Type de point de terminaison de test utilisé pour tester des appareils avec Device Advisor. Pour de plus amples informations, veuillez consulter ???.
AWS IoT Core data beta (aperçu)	`iot:Data-Beta`	Type de point de terminaison réservé aux versions bêta. Pour plus d'informations sur son utilisation actuelle, consultez ???.

Vous pouvez également utiliser votre propre nom de domaine complet (FQDN), tel que*example.com*, et le certificat de serveur associé pour connecter des appareils AWS IoT en utilisant. the section called “Configurations de domaine”

AWS IoT Device SDKs

L' AWS IoT appareil vous SDKs aide à connecter vos appareils IoT AWS IoT Core et prend en charge les protocoles MQTT et MQTT sur WSS.

L' AWS IoT appareil SDKs diffère de l' AWS IoT appareil AWS SDKs SDKs en ce sens qu'il répond aux besoins de communication spécialisés des appareils IoT, mais ne prend pas en charge tous les services pris en charge par le AWS SDKs. Les AWS IoT appareils SDKs sont compatibles avec ceux AWS SDKs qui prennent en charge tous les AWS services ; cependant, ils utilisent différentes méthodes d'authentification et se connectent à différents points de terminaison, ce qui peut rendre leur utilisation AWS SDKs peu pratique sur un appareil IoT.

Appareils mobiles

Ils the section called “AWS Applications mobiles SDKs” prennent en charge à la fois les communications des appareils MQTT APIs, une partie du AWS IoT service et APIs d'autres AWS services. Si vous développez sur un appareil mobile compatible, consultez son SDK pour voir s'il s'agit de la meilleure option pour développer votre solution IoT.

C++

AWS IoT SDK pour appareils C++

Le SDK AWS IoT C++ Device permet aux développeurs de créer des applications connectées en utilisant AWS et en utilisant APIs les AWS IoT Core services. Ce kit SDK a été conçu en particulier pour les appareils qui ne sont pas limités en ressources et qui nécessitent des fonctions avancées, telles que la mise en file d'attente des messages, la prise en charge du multithreading et les dernières fonctions de langue. Pour plus d’informations, consultez les ressources suivantes :

- AWS IoT SDK C++ v2 de l'appareil activé GitHub
- AWS IoT Readme du SDK C++ v2 de l'appareil
- AWS IoT Exemples du SDK C++ v2 pour appareils
- AWS IoT Documentation de l'API C++ v2 du SDK pour appareils

Python

AWS IoT SDK de périphérique pour Python

Le AWS IoT Device SDK for Python permet aux développeurs d'écrire des scripts Python afin d'utiliser leurs appareils pour accéder à la AWS IoT plateforme via le protocole MQTT ou MQTT over the WebSocket Secure (WSS). En connectant leurs appareils aux AWS IoT Core services, les utilisateurs peuvent travailler en toute sécurité avec le courtier de messages, les règles et le service Device Shadow qui AWS IoT Core fournit AWS Lambda, ainsi qu'avec d'autres AWS services tels qu'Amazon Kinesis et Amazon S3, etc. APIs

- AWS IoT SDK de périphérique pour Python v2 sur GitHub
- AWS IoT SDK du périphérique pour Python v2 Readme
- AWS IoT Exemples de SDK de périphérique pour Python v2
- AWS IoT Documentation de l'API du SDK du périphérique pour Python v2

JavaScript

AWS IoT SDK de périphérique pour JavaScript

Le AWS IoT Device SDK pour JavaScript permet aux développeurs d'écrire des JavaScript applications qui y accèdent APIs AWS IoT Core en utilisant MQTT ou MQTT via le protocole. WebSocket Il peut être utilisé dans des environnements Node.js et des applications de navigateur. Pour plus d'informations, consultez les ressources suivantes :

- AWS IoT SDK de l'appareil pour JavaScript v2 sur GitHub
- AWS IoT SDK du périphérique pour Readme JavaScript v2
- AWS IoT Exemples de SDK du périphérique pour la JavaScript version v2
- AWS IoT Documentation de l'API du SDK de l'appareil pour la JavaScript version v2

Java

AWS IoT SDK du périphérique pour Java

Le AWS IoT Device SDK for Java permet aux développeurs Java d'accéder APIs au via MQTT ou MQTT via AWS IoT Core le protocole. WebSocket Le SDK prend en charge le service Device Shadow. Vous pouvez accéder au service Shadows à l'aide des méthodes HTTP, notamment GET, UPDATE et DELETE. Le kit SDK prend également en charge un modèle d'accès aux shadows simplifié, qui permet aux développeurs d'échanger des données avec des shadows en utilisant des méthodes getter et setter, sans avoir à sérialiser ou désérialiser des documents JSON. Pour plus d'informations, consultez les ressources suivantes :

- AWS IoT SDK de périphérique pour Java v2 activé GitHub
- AWS IoT SDK du périphérique pour Java v2 Readme
- AWS IoT Exemples de SDK pour appareils pour Java v2
- AWS IoT Documentation de l'API Device SDK for Java v2

Embedded C

AWS IoT SDK de périphérique pour Embedded C

> **Important**
>
> Ce SDK est destiné à être utilisé par des développeurs de logiciels embarqués expérimentés.

Le Kit SDK des appareils AWS IoT pour Embedded C (C-SDK) est une collection de fichiers source C sous licence open source du MIT qui peuvent être utilisés dans des applications intégrées pour connecter en toute sécurité des appareils IoT à AWS IoT Core. Il inclut les bibliothèques MQTT, JSON Parser et AWS IoT Device Shadow, entre autres. Il est distribué sous forme source et destiné à être intégré au micrologiciel client avec le code d'application, d'autres bibliothèques et, en option, un RTOS (Real Time Operating System).

Kit SDK des appareils AWS IoT pour Embedded C Il est généralement destiné aux appareils aux ressources limitées qui nécessitent un environnement d'exécution optimisé en langage C. Vous pouvez utiliser le SDK sur n'importe quel système d'exploitation et l'héberger sur n'importe quel type de processeur (par exemple, MCUs et MPUs). Si votre appareil dispose de suffisamment de mémoire et de ressources de traitement, nous vous recommandons d'utiliser l'un des autres AWS IoT appareils et appareils mobiles SDKs, tels que le SDK pour le AWS IoT périphérique pour C++ JavaScript, Java ou Python.

Pour plus d’informations, consultez les ressources suivantes :

- AWS IoT SDK du périphérique pour Embedded C on GitHub
- AWS IoT SDK du périphérique pour Embedded C Readme
- AWS IoT SDK de périphérique pour les échantillons C intégrés

Protocoles de communication des appareils

AWS IoT Core prend en charge les appareils et les clients qui utilisent les protocoles MQTT et MQTT over WebSocket Secure (WSS) pour publier des messages et s'y abonner, ainsi que les appareils et clients qui utilisent le protocole HTTPS pour publier des messages. Tous les protocoles prennent en charge IPv4 et IPv6. Cette section décrit les différentes options de connexion pour les appareils et les clients.

Versions du protocole TLS

AWS IoT Core utilise TLS version 1.2 et TLS version 1.3 pour chiffrer toutes les communications. Vous pouvez configurer des versions de politique TLS supplémentaires pour votre point de terminaison en configurant les paramètres TLS dans les configurations de domaine. Lorsqu'ils connectent des appareils AWS IoT Core, les clients peuvent envoyer l'extension SNI (Server Name Indication), qui est requise pour des fonctionnalités telles que l'enregistrement multi-comptes, les points de terminaison configurables, les domaines personnalisés et les points de terminaison VPC. Pour de plus amples informations, veuillez consulter Sécurité du transport dans AWS IoT.

Les AWS IoT Device SDKs prennent en charge le MQTT et le MQTT sur WSS et répondent aux exigences de sécurité des connexions client. Nous vous recommandons d'utiliser le AWS IoT Device SDKs pour connecter les clients à AWS IoT.

Protocols, port mappings, and authentication (Protocoles, mappages de ports et authentification)

La façon dont un appareil ou un client se connecte au courtier de messages est configurable à l'aide d'un type d'authentification. Par défaut ou lorsqu'aucune extension SNI n'est envoyée, la méthode d'authentification est basée sur le protocole d'application, le port et l'extension TLS ALPN (Application Layer Protocol Negotiation) utilisés par les appareils. Le tableau suivant répertorie l'authentification attendue en fonction du port, du port et de l'ALPN.

Protocoles, authentification et mappages de port

Protocole	Opérations prises en charge	Authentification	Port	Nom du protocole ALPN
MQTT terminé WebSocket	Publier, S'abonner	Signature Version 4	443	N/A
MQTT terminé WebSocket	Publier/s 'abonner	Authentification personnalisée	443	N/A
MQTT	Publier, S'abonner	Certificat de client X.509	443†	`x-amzn-mqtt-ca`
MQTT	Publier, S'abonner	Certificat de client X.509	8883	N/A

Protocole	Opérations prises en charge	Authentification	Port	Nom du protocole ALPN
MQTT	Publier/s 'abonner	Authentification personnalisée	443†	`mqtt`
HTTPS	Publier uniquement	Signature Version 4	443	N/A
HTTPS	Publier uniquement	Certificat de client X.509	443†	`x-amzn-http-ca`
HTTPS	Publier uniquement	Certificat de client X.509	8443	N/A
HTTPS	Publier uniquement	Authentification personnalisée	443	N/A

Application Layer Protocol Negotiation (ALPN)

† Lors de l'utilisation de configurations de point de terminaison par défaut, les clients qui se connectent sur le port 443 avec l'authentification par certificat client X.509 doivent implémenter l'extension TLS ALPN (Application Layer Protocol Negotiation) et utiliser le nom du protocole ALPN indiqué dans l'ALPN ProtocolNameList envoyé par le client dans le cadre du message. `ClientHello`
Sur le port 443, le point de terminaison IoT:Data-ATS prend en charge le x-amzn-http-ca protocole HTTP ALPN, mais pas le point de terminaison IoT:Jobs.
Sur le port 8443 HTTPS et le port 443 MQTT avec ALPN x-amzn-mqtt-ca, l'authentification personnalisée ne peut pas être utilisée.

Les clients se connectent aux terminaux Compte AWS de leurs appareils. Consultez la section the section called “AWS IoT données de l'appareil et points de terminaison de service” pour plus d'informations sur la façon de trouver les points de terminaison de l'appareil de votre compte.

> **Note**
>
> AWS SDKs ne nécessitent pas l'URL complète. Ils ont uniquement besoin du nom d'hôte du point de terminaison, tel que l'`pubsub.py`exemple du AWS IoT Device SDK for Python on. GitHub La transmission de l'URL complète comme indiqué dans le tableau suivant peut générer une erreur telle qu'un nom d'hôte non valide.

Connexion à AWS IoT Core

Protocole	Point de terminaison ou URL
MQTT	*iot-endpoint*
MQTT via WSS	wss://*iot-endpoint* /mqtt
HTTPS	https://*iot-endpoint* /topics

Choix d'un protocole d'application pour la communication de votre appareil

Pour la plupart des communications entre appareils IoT via les terminaux de l'appareil, vous devez utiliser les protocoles Secure MQTT ou MQTT over WebSocket Secure (WSS) ; toutefois, les points de terminaison des appareils prennent également en charge le protocole HTTPS.

Le tableau suivant compare l' AWS IoT Core utilisation des deux protocoles de haut niveau (MQTT et HTTPS) pour la communication entre appareils.

AWS IoT protocoles des appareils (MQTT et HTTPS) side-by-side

Fonctionnalité	MQTT	HTTPS
Assistance pour la publication/abonnement	Publication et d'abonnement	Publier uniquement
Prise en charge de SDK	AWS L'appareil SDKs prend en charge les protocoles MQTT et WSS	Pas de prise en charge du SDK, mais vous pouvez utiliser des méthodes spécifiques à la langue pour effectuer des requêtes HTTPS

Fonctionnalité	MQTT	HTTPS
Prise en charge de la qualité de service	QoS MQTT niveaux 0 et 1	La QoS est prise en charge en transmettant un paramètre de chaîne de requête `?qos=qos` dont la valeur peut être 0 ou 1. Vous pouvez ajouter cette chaîne de requête pour publier un message avec la valeur de QoS souhaitée.
Les messages de réception peuvent-ils être manqués lorsque l'appareil était hors ligne	Oui	Non
Soutien sur le terrain `clientId`	Oui	Non
Détection de déconnexion de l'appareil	Oui	Non
Communications sécurisées	Oui. Consultez ???.	Oui. Consultez ???.
Définitions de rubriques	Application définie	Application définie
Format des données des messages	Application définie	Application définie
Surcharge de protocole	Inférieur	Supérieur
Consommation d’énergie	Inférieur	Supérieur

Choix d'un type d'authentification pour les communications avec votre appareil

Vous pouvez configurer le type d'authentification pour votre point de terminaison IoT à l'aide de points de terminaison configurables. Vous pouvez également utiliser la configuration par défaut et déterminer comment vos appareils s'authentifient à l'aide du protocole d'application, du port et de la

combinaison d'extensions TLS ALPN. Le type d'authentification que vous choisissez détermine la manière dont vos appareils s'authentifieront lors de la connexion. AWS IoT Core Il existe cinq types d'authentification :

un certificat X.509

Authentifiez les appareils à l'aide de certificats clients X.509, qui AWS IoT Core valident l'authentification du périphérique. Ce type d'authentification fonctionne avec les protocoles Secure MQTT (MQTT over TLS) et HTTPS.

Certificat X.509 avec autorisateur personnalisé

Authentifiez les appareils à l'aide de certificats client X.509 et effectuez des actions d'authentification supplémentaires à l'aide d'un autorisateur personnalisé, qui recevra les informations du certificat client X.509. Ce type d'authentification fonctionne avec les protocoles Secure MQTT (MQTT over TLS) et HTTPS. Ce type d'authentification n'est possible qu'à l'aide de points de terminaison configurables avec l'authentification personnalisée X.509. Il n'existe pas d'option ALPN.

AWS Version 4 de la signature (SigV4)

Authentifiez les appareils à l'aide de Cognito ou de votre service principal, en soutenant la fédération sociale et d'entreprise. Ce type d'authentification fonctionne avec les protocoles MQTT over WebSocket Secure (WSS) et HTTPS.

Autorisateur personnalisé

Authentifiez les appareils en configurant une fonction Lambda pour traiter les informations d'authentification personnalisées envoyées à. AWS IoT Core Ce type d'authentification fonctionne avec les protocoles Secure MQTT (MQTT over TLS), HTTPS et MQTT over WebSocket Secure (WSS).

Par défaut

Authentifiez les appareils en fonction de l'extension ALPN (Port and/or Application Layer Protocol Negotiation) utilisée par les appareils. Certaines options d'authentification supplémentaires ne sont pas prises en charge. Pour de plus amples informations, veuillez consulter ???.

Le tableau ci-dessous présente toutes les combinaisons de types d'authentification et de protocoles d'application prises en charge.

Combinaisons prises en charge de types d'authentification et de protocoles d'application

Type d’authentification	MQTT sécurisé (MQTT sur TLS)	MQTT sur WebSocket Secure (WSS)	HTTPS	Par défaut
un certificat X.509	✓		✓	
Certificat X.509 avec autorisateur personnalisé	✓		✓	
AWS Version 4 de la signature (SigV4)		✓	✓	
Autorisateur personnalisé	✓	✓	✓	
Par défaut	✓			✓

Limites de durée de connexion

Il n'est pas garanti que les connexions HTTPS durent plus longtemps que le temps nécessaire pour recevoir et répondre aux demandes.

La durée de connexion MQTT dépend de la fonctionnalité d'authentification que vous utilisez. Le tableau suivant répertorie la durée maximale de connexion dans des conditions idéales pour chaque fonctionnalité.

Durée de connexion MQTT par fonction d'authentification

Fonctionnalité	Durée maximum *
Certificat de client X.509	1 à 2 semaines
Authentification personnalisée	1 à 2 semaines
Signature Version 4	Jusqu'à 24 heures

* Non garanti

Avec les certificats X.509 et l'authentification personnalisée, la durée de connexion n'est pas limitée, mais elle peut être aussi courte que quelques minutes. Ces erreurs peuvent se produire pour diverses raisons. La liste suivante contient certaines des raisons les plus courantes.

- Interruptions de disponibilité du Wi-Fi
- Interruptions de connexion du fournisseur de services Internet (FSI)
- Correctifs de service
- Déploiements de services
- Service Auto Scaling
- Service hôte non disponible
- Problèmes et mises à jour de l'équilibreur de charge
- Erreurs côté du client

Vos appareils doivent mettre en œuvre des stratégies pour détecter les déconnexions et les reconnexions. Pour plus d'informations sur les événements de déconnexion et des conseils sur la façon de les gérer, consultez ??? à???.

MQTT

Le MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie léger et largement adopté, conçu pour les appareils restreints. AWS IoT Core le soutient pour MQTT est basé sur les spécifications MQTT v3.1.1 et MQTT v5.0, avec quelques différences, comme indiqué dans. the section called “AWS IoT différences par rapport aux spécifications MQTT” En tant que dernière version de la norme, MQTT 5 introduit plusieurs fonctionnalités clés qui renforcent la robustesse d’un système basé sur MQTT, notamment de nouvelles améliorations en termes de capacité de mise à l’échelle, un meilleur signalement des erreurs avec les réponses au code de motif, les délais d’expiration des messages et des sessions, et des en-têtes de message utilisateur personnalisés. Pour plus d'informations sur les fonctionnalités prises en charge par MQTT 5, voir Fonctionnalités prises AWS IoT Core en charge par MQTT 5. AWS IoT Core prend également en charge la communication entre les versions MQTT (MQTT 3 et MQTT 5). Un diffuseur de publication MQTT 3 peut envoyer un message MQTT 3 à un abonné MQTT 5 qui recevra un message de publication MQTT 5, et vice versa.

AWS IoT Core prend en charge les connexions de périphériques qui utilisent le protocole MQTT et le protocole MQTT over WSS et qui sont identifiées par un identifiant client. Les AWS IoT Device

SDKs prennent en charge les deux protocoles et sont les méthodes recommandées pour connecter les appareils AWS IoT Core. L' AWS IoT appareil SDKs prend en charge les fonctions nécessaires aux appareils et aux clients pour se connecter aux AWS IoT services et y accéder. L'appareil SDKs prend en charge les protocoles d'authentification requis par les AWS IoT services et les exigences d'identification de connexion requises par le protocole MQTT et les protocoles MQTT over WSS. Pour plus d'informations sur la façon de se connecter AWS IoT à l' AWS appareil SDKs et pour obtenir des liens vers des exemples de AWS IoT langues prises en charge, consultezthe section called "Connexion à MQTT à l'aide de l'appareil AWS IoT SDKs". Pour plus d'informations sur les méthodes d'authentification et de mappage de ports pour les messages MQTT, consultez ???.

Bien que nous vous recommandions d'utiliser l' AWS IoT appareil SDKs pour vous y connecter AWS IoT, ils ne sont pas obligatoires. Si vous n'utilisez pas l' AWS IoT appareil SDKs, vous devez toutefois fournir la sécurité de connexion et de communication nécessaire. Les clients doivent envoyer l'extension TLS SNI (Server Name Indication) dans la demande de connexion. Les tentatives de connexion qui n'incluent pas le SNI sont refusées. Pour plus d'informations, consultez la section Sécurité du transport dans AWS IoT. Les clients qui utilisent des utilisateurs et des AWS informations d'identification IAM pour authentifier les clients doivent fournir l'authentification Signature Version 4 correcte.

Dans cette rubrique :

- Connexion à MQTT à l'aide de l'appareil AWS IoT SDKs
- Options de qualité de service (QoS) MQTT
- Sessions permanentes MQTT
- MQTT a retenu les messages
- Messages MQTT Last Will and Testament (LWT)
- Utilisation de ConnectAttributes
- Fonctionnalités prises en charge par MQTT 5
- Propriétés MQTT 5
- Codes de motif MQTT
- AWS IoT différences par rapport aux spécifications MQTT

Connexion à MQTT à l'aide de l'appareil AWS IoT SDKs

Cette section contient des liens vers le AWS IoT périphérique SDKs et vers le code source d'exemples de programmes illustrant comment connecter un appareil à celui-ci AWS IoT. Les

exemples d'applications liés ici montrent comment se connecter à AWS IoT l'aide du protocole MQTT et de MQTT via WSS.

> **Note**
>
> L' AWS IoT appareil SDKs a publié un client MQTT 5.

C++

Utilisation du SDK de périphériques AWS IoT C++ pour connecter des appareils

- Code source d'un exemple d'application qui montre un exemple de connexion MQTT en C++
- AWS IoT SDK du périphérique pour C++ v2 sur GitHub

Python

Utilisation du AWS IoT Device SDK pour Python pour connecter des appareils

- Code source d'un exemple d'application qui montre un exemple de connexion MQTT en Python
- AWS IoT Device SDK v2 pour Python sur GitHub

JavaScript

Utilisation du AWS IoT Device SDK pour JavaScript connecter des appareils

- Code source d'un exemple d'application qui montre un exemple de connexion MQTT dans JavaScript
- AWS IoT SDK de l'appareil pour la JavaScript version 2 sur GitHub

Java

Utilisation du AWS IoT Device SDK for Java pour connecter des appareils

> **Note**
>
> Le AWS IoT Device SDK for Java v2 prend désormais en charge le développement Android. Pour plus d'informations, consultez AWS IoT Device SDK for Android.

- Code source d'un exemple d'application qui montre un exemple de connexion MQTT en Java
- AWS IoT SDK de périphérique pour Java v2 activé GitHub

Embedded C

Utilisation du AWS IoT Device SDK for Embedded C pour connecter des appareils

> **Important**
>
> Ce SDK est destiné à être utilisé par des développeurs de logiciels embarqués expérimentés.

- Code source d'un exemple d'application qui montre un exemple de connexion MQTT dans Embedded C
- AWS IoT SDK du périphérique pour Embedded C on GitHub

Options de qualité de service (QoS) MQTT

AWS IoT et le AWS IoT périphérique SDKs prend en charge les niveaux de qualité de service (QoS) 0 MQTT et. 1 Le protocole MQTT définit un troisième niveau de QoS, le 2 niveau, AWS IoT mais ne le supporte pas. Seul le protocole MQTT prend en charge la fonctionnalité QoS. HTTPS prend en charge la QoS en transmettant un paramètre de chaîne de requête `?qos=qos` dont la valeur peut être 0 ou 1.

Ce tableau décrit comment chaque niveau de QoS affecte les messages publiés par et vers l'agent de messages.

Avec un niveau de QoS de...	Le message est…	Commentaires
QoS niveau 0	Envoyé zéro fois ou plus	Ce niveau doit être utilisé pour les messages envoyés via des liens de communication fiables ou qui peuvent être manqués sans problème.
QoS niveau 1	Envoyé au moins une fois, puis à plusieurs reprises	Le message n'est pas considéré comme complet tant

Avec un niveau de QoS de...	Le message est…	Commentaires
	jusqu'à ce qu'une réponse PUBACK soit reçue	que l'expéditeur n'a pas reçu de réponse PUBACK indiquant que la livraison a été réussie.

Sessions permanentes MQTT

Les sessions permanentes stockent les abonnements et les messages d'un client, avec une qualité de service (QoS) de 1, qui n'ont pas été reconnus par le client. Lorsque l'appareil se reconnecte à une session permanente, la session reprend, les abonnements sont rétablis et les messages d'abonnement reçus et stockés sans accusé de réception avant la reconnexion sont envoyés au client.

Le traitement des messages enregistrés est enregistré dans CloudWatch et CloudWatch Logs. Pour plus d'informations sur les entrées écrites dans CloudWatch et dans CloudWatch les journaux, reportez-vous aux Métriques d'agent de messages sections etEntrée de journal en file d'attente.

Création d'une session permanente

Dans MQTT 3, vous créez une session permanente MQTT en envoyant un message `CONNECT` et en définissant l'indicateur `cleanSession` sur `0`. Si aucune session n'existe pour le client qui envoie le message `CONNECT`, une nouvelle session permanente est créée. Si une session existe déjà pour le client, celui-ci la reprend. Pour créer une session propre, vous envoyez un message `CONNECT` et vous définissez l'indicateur `cleanSession` sur `1`, et l'agent ne stockera aucun état de session lorsque le client se déconnecte.

Dans MQTT 5, vous gérez les sessions permanentes en définissant l'indicateur `Clean Start` et `Session Expiry Interval`. Le démarrage propre contrôle le début de la session de connexion et la fin de la session précédente. Lorsque vous définissez `Clean Start =1`, une nouvelle session est créée et une session précédente est interrompue si elle existe. Lorsque vous définissez `Clean Start =0`, la session de connexion reprend une session précédente si elle existe. L'intervalle d'expiration de session contrôle la fin de la session de connexion. L'intervalle d'expiration de session indique le temps, en secondes (entier de 4 octets), pendant lequel une session persistera après la déconnexion. Le paramètre `Session Expiry interval = 0` entraîne la fin de la session immédiatement après la déconnexion. Si l'intervalle d'expiration de session n'est pas spécifié dans le message CONNECT, la valeur par défaut est 0.

Démarrage propre MQTT 5 et expiration de session

Valeur de la propriété	Description
`Clean Start= 1`	Crée une nouvelle session et met fin à une session précédente s'il en existe une.
`Clean Start= 0`	Reprend une session s'il en existe une précédente.
`Session Expiry Interval> 0`	Maintient une session.
`Session Expiry interval= 0`	Ne fait pas perdurer une session.

Dans MQTT 5, si vous définissez `Clean Start = 1` et `Session Expiry Interval =0`, cela équivaut à une session propre de MQTT 3. Si vous définissez `Clean Start = 0` et `Session Expiry Interval >0`, cela équivaut à une session permanente MQTT 3.

> **Note**
>
> Les sessions permanentes de plusieurs versions MQTT (MQTT 3 et MQTT 5) ne sont pas prises en charge. Une session permanente MQTT 3 ne peut pas être reprise en tant que session MQTT 5, et vice versa.

Opérations au cours d'une session permanente

Les clients utilisent l'attribut `sessionPresent` dans le message de connexion reconnue (`CONNACK`), afin de déterminer si une session permanente est présente. Si `sessionPresent` est `1`, une session permanente est présente et tous les messages stockés pour le client sont remis au client une fois que celui-ci a reçu `CONNACK`, comme décrit dans Trafic de messages après reconnexion à une session permanente. Si `sessionPresent` est `1`, le client n'a pas besoin de se réabonner. Toutefois, si `sessionPresent` est défini sur `0`, aucune session permanente n'est présente et le client doit se réabonner à ses filtres de rubrique.

Une fois que le client rejoint la session permanente, il peut publier des messages et à s'abonner aux filtres de rubrique sans aucun indicateur supplémentaire sur chaque opération.

Trafic de messages après reconnexion à une session permanente

Une session permanente représente une connexion continue entre un client et un agent de messages MQTT. Lorsqu'un client se connecte à l'agent de messages à l'aide d'une session permanente, ce dernier enregistre tous les abonnements souscrits par le client pendant la connexion. Lorsque le client se déconnecte, le courtier de messages conserve les messages d'une QoS 1 et les nouveaux messages d'une QoS 1 publiés sur les rubriques auxquelles le client s'est abonné. Les messages sont stockés conformément à la limite du compte. Les messages dépassant cette limite seront supprimés. Pour plus d'informations sur les limites de messages permanents, veuilelz consulter AWS IoT Core quotas de point de terminaison. Lorsque le client se reconnecte à sa session permanente, tous les abonnements sont réactivés et tous les messages stockés sont envoyés au client à un rythme maximum de 10 messages par seconde. Dans MQTT 5, si un QoS1 sortant avec l'intervalle d'expiration des messages expire alors qu'un client est hors ligne, une fois la connexion rétablie, le client ne recevra pas le message expiré.

Après la reconnexion, les messages stockés sont envoyés au client, à un débit limité à 10 messages stockés par seconde, ainsi que tout le trafic de messages en cours jusqu'à ce que la limite `Publish requests per second per connection` soit atteinte. Le taux de livraison des messages stockés étant limité, la remise de tous les messages stockés prendra plusieurs secondes si une session comporte plus de 10 messages stockés à remettre après la reconnexion.

Fin d'une session permanente

Les sessions permanentes peuvent prendre fin de différentes manières :

- Le délai d'expiration de la session permanente est expiré. Le délai d'expiration de session permanente démarre lorsque l'agent de messages détecte qu'un client s'est déconnecté, soit en raison de la déconnexion du client, soit en raison de l'expiration du délai de connexion.
- Le client envoie un message `CONNECT` qui définit l'indicateur `cleanSession` sur `1`.

Dans MQTT 3, la valeur par défaut du délai d'expiration des sessions permanentes est d'une heure, et cela s'applique à toutes les sessions du compte.

Dans MQTT 5, vous pouvez définir l'intervalle d'expiration de session pour chaque session sur les paquets CONNECT et DISCONNECT.

Pour l'intervalle d'expiration de session sur le paquet DISCONNECT :

- Si la session en cours a un intervalle d'expiration de session de 0, vous ne pouvez pas définir un intervalle d'expiration de session supérieur à 0 sur le paquet DISCONNECT.
- Si la session en cours a un intervalle d'expiration de session supérieur à 0 et que vous définissez l'intervalle d'expiration de session sur 0 sur le paquet DISCONNECT, la session sera terminée sur DISCONNECT.
- Sinon, l'intervalle d'expiration de session sur le paquet DISCONNECT mettra à jour l'intervalle d'expiration de session de la session en cours.

> **Note**
>
> Les messages stockés en attente d'être envoyés au client à la fin d'une session sont supprimés ; cependant, ils sont toujours facturés au tarif de messagerie standard, même s'ils n'ont pas pu être envoyés. Pour plus d'informations sur la tarification des messages, consultez AWS IoT Core Tarification. Vous pouvez configurer l'intervalle d'expiration.

Reconnexion après expiration d'une session permanente

Si un client ne se reconnecte pas à sa session permanente avant son expiration, celle-ci se termine et les messages enregistrés sont supprimés. Lorsqu'un client se reconnecte après l'expiration de la session et définit un indicateur `cleanSession` sur `0`, le service crée une nouvelle session persistante. Les abonnements ou messages de la session précédente ne sont pas disponibles pour cette session car ils ont été supprimés à l'expiration de la session précédente.

Frais liés aux messages de session persistants

Les messages vous sont facturés Compte AWS lorsque le courtier de messages envoie un message à un client ou lors d'une session permanente hors ligne. Lorsqu'un appareil hors ligne doté d'une session permanente se reconnecte et reprend sa session, les messages enregistrés sont transmis à l'appareil et débités de nouveau sur votre compte. Pour plus d'informations sur la tarification des messages, consultez la section AWS IoT Core Tarification - Messagerie.

Le délai d'expiration des sessions permanentes par défaut d'une heure peut être augmenté en utilisant le processus d'augmentation de limite standard. Notez que l'augmentation du délai d'expiration de la session peut augmenter le coût de vos messages, car ce délai supplémentaire pourrait permettre de stocker davantage de messages sur l'appareil hors ligne et ces messages supplémentaires seraient débités de votre compte au tarif de messagerie standard. L'heure d'expiration de la session est approximative et une session peut persister jusqu'à 30 minutes de plus

que la limite du compte ; toutefois, une session ne sera pas inférieure à la limite du compte. Pour de plus amples informations sur les limites de sessions, veuillez consulter AWS Service Quotas.

MQTT a retenu les messages

AWS IoT Core supporte l'indicateur RETAIN décrit dans le protocole MQTT. Lorsqu'un client définit l'indicateur RETAIN sur un message MQTT qu'il publie, il AWS IoT Core enregistre le message. Il peut ensuite être envoyé aux nouveaux abonnés, récupéré en appelant l'opération `GetRetainedMessage` et visualisé dans la AWS IoT console.

Exemples d'utilisation de messages retenus au format MQTT

- En tant que message de configuration initiale

  Les messages retenus au format MQTT sont envoyés à un client une fois que celui-ci s'est abonné à une rubrique. Si vous souhaitez que tous les clients abonnés à un sujet reçoivent le message MQTT conservé juste après leur inscription, vous pouvez publier un message de configuration avec l'indicateur RETAIN activé. Les clients abonnés reçoivent également des mises à jour de cette configuration chaque fois qu'un nouveau message de configuration est publié.
- En tant que dernier message d'état connu

  Les appareils peuvent définir l'indicateur RETAIN sur les messages d'état actuel afin que AWS IoT Core les enregistre. Lorsque les applications se connectent ou se reconnectent, elles peuvent s'abonner à cette rubrique et obtenir le dernier état signalé juste après s'être abonnées à la rubrique de message conservée. De cette façon, ils peuvent éviter d'avoir à attendre le prochain message de l'appareil pour voir l'état actuel.

Dans cette section :

- Tâches courantes avec des messages retenus au format MQTT dans AWS IoT Core
- Facturation et messages retenus
- Comparaison des messages MQTT retenus et des sessions permanentes MQTT
- MQTT a conservé les messages et AWS IoT Device Shadows

Tâches courantes avec des messages retenus au format MQTT dans AWS IoT Core

AWS IoT Core enregistre les messages MQTT avec l'indicateur RETAIN activé. Ces messages retenus sont envoyés à tous les clients abonnés au sujet, sous la forme d'un message MQTT normal, et ils sont également stockés pour être envoyés aux nouveaux abonnés à la rubrique.

Les messages retenus au format MQTT nécessitent des actions politiques spécifiques pour autoriser les clients à y accéder. Pour des exemples d'utilisation des politiques relatives aux messages retenus, consultez Exemples de stratégies de messages conservés.

Cette section décrit les opérations courantes impliquant les messages retenus.

- Création d'un message retenu

  Le client détermine si un message est retenu lorsqu'il publie un message MQTT. Les clients peuvent définir l'indicateur RETAIN lorsqu'ils publient un message à l'aide d'un SDK des appareils. Les applications et les services peuvent définir l'indicateur RETAIN lorsqu'ils utilisent l'`Publish`action pour publier un message MQTT.

  Un seul message par nom de rubrique est retenu. Un nouveau message dont l'indicateur RETAIN est défini et publié dans une rubrique remplace tout message retenu existant qui a été envoyé au sujet précédemment.

  REMARQUE : Vous ne pouvez pas publier dans une rubrique réservée lorsque l'indicateur RETAIN est activé.
- Abonnement à un sujet de message retenu

  Les clients s'abonnent aux rubriques de message retenus comme ils le feraient pour n'importe quel autre rubrique de message MQTT. L'indicateur RETAIN est activé pour les messages retenus reçus en vous abonnant à un sujet de message retenu.

  Les messages conservés sont supprimés AWS IoT Core lorsqu'un client publie un message conservé avec une charge utile de 0 octet dans le sujet du message conservé. Les clients qui se sont abonnés à la rubrique du message retenu recevront également le message de 0 octet.

  L'abonnement à un filtre de rubrique générique qui inclut un sujet de message retenu permet au client de recevoir les messages suivants publiés dans le sujet du message retenu, mais il ne transmet pas le message retenu lors de l'abonnement.

  REMARQUE : Pour recevoir un message retenu lors de l'abonnement, le filtre de rubrique de la demande d'abonnement doit correspondre exactement à la rubrique du message retenu.

  L'indicateur RETAIN est activé pour les messages retenus reçus lors de l'abonnement à une rubrique de message retenu. Les messages retenus qui sont reçus par un client abonné après son abonnement ne le sont pas.
- Récupération d'un message retenu

Les messages retenus sont remis automatiquement aux clients lorsqu'ils s'abonnent à la rubrique contenant le message retenu. Pour qu'un client reçoive le message retenu lors de son abonnement, il doit s'abonner au nom exact de rubrique du message retenu. L'abonnement à un filtre de rubrique générique qui inclut une rubrique de message retenu permet au client de recevoir les messages suivants publiés dans la rubrique du message retenu, mais il ne transmet pas le message retenu lors de l'abonnement.

Les services et applications peuvent répertorier et récupérer les messages retenus en appelant `ListRetainedMessages` et `GetRetainedMessage`.

Aucun client n'est empêché de publier des messages dans une rubrique de message retenu sans définir l'indicateur RETAIN. Cela peut entraîner des résultats inattendus, tels que le message retenu ne correspondant pas au message reçu en vous abonnant à la rubrique.

Avec MQTT 5, si l'intervalle d'expiration d'un message retenu est défini et que le message retenu expire, un nouvel abonné qui s'abonne à cette rubrique ne recevra pas le message retenu une fois son abonnement réussi.

- Répertorier les sujets des messages retenus

  Vous pouvez répertorier les messages retenus en appelant `ListRetainedMessages` et les messages retenus peuvent être consultés dans la AWS IoT console.

- Obtenir les détails des messages retenus

  Vous pouvez obtenir les détails des messages retenus en appelant `GetRetainedMessage` et ils peuvent être consultés dans la AWS IoT console.

- Retenir un message volontaire

  Les messages MQTT Will créés lorsqu'un appareil se connecte peuvent être retenus en définissant l'indicateur `Will Retain` dans le champ `Connect Flag bits`.

- Supprimer un message retenu

  Les appareils, les applications et les services peuvent supprimer un message retenu en publiant un message avec l'indicateur RETAIN activé et une charge utile de message vide (0 octet) dans le nom du sujet du message retenu à supprimer. Ces messages suppriment le message conservé de AWS IoT Core, sont envoyés aux clients abonnés au sujet, mais ils ne sont pas conservés par AWS IoT Core.

Les messages retenus peuvent également être supprimés de manière interactive en accédant au message retenu dans la AWS IoT console. Les messages retenus supprimés à l'aide de la AWS IoT console envoient également un message de 0 octet aux clients abonnés au sujet du message retenu.

Les messages retenus ne peuvent pas être restaurés après leur suppression. Un client devra publier un nouveau message retenu pour remplacer le message supprimé.

- Débogage et résolution des problèmes liés aux messages retenus

  La AWS IoT console fournit plusieurs outils pour vous aider à résoudre les problèmes liés aux messages retenus :

  - La page des messages retenus

    La page Messages retenus de la console AWS IoT fournit une liste paginée des messages retenus qui ont été stockés par votre compte dans la région actuelle. À partir de cette page, vous pouvez :

    - Consultez les détails de chaque message retenu, tels que la charge utile du message, la QoS, l'heure à laquelle il a été reçu.
    - Mettez à jour le contenu d'un message retenu.
    - Supprimez un message retenu.
  - Le client de test MQTT

    La page du client de test MQTT de la console AWS IoT permet de s'abonner et de publier sur des sujets MQTT. L'option de publication vous permet de définir l'indicateur RETAIN sur les messages que vous publiez afin de simuler le comportement de vos appareils.

  Certains résultats inattendus peuvent être le résultat de ces aspects de la manière dont les messages conservés sont implémentés dans AWS IoT Core.

  - Limit es de messages retenues

    Lorsqu'un compte a stocké le nombre maximum de messages conservés, AWS IoT Core renvoie une réponse limitée aux messages publiés avec l'option RETAIN définie et à des charges utiles supérieures à 0 octet jusqu'à ce que certains messages conservés soient supprimés et que le nombre de messages conservés tombe en dessous de la limite.
  - Ordre de distribution des messages retenus

    La séquence d'envoi des messages retenus et des messages souscrits n'est pas garantie.

Facturation et messages retenus

La publication de messages avec l'indicateur RETAIN activé par un client, à l'aide de la console AWS IoT ou par appel `Publish` entraîne des frais de messagerie supplémentaires décrits dans la section AWS IoT Core Tarification - Messagerie.

La récupération des messages conservés par un client, à l'aide de AWS IoT la console ou par appel `GetRetainedMessage`entraîne des frais de messagerie en plus des frais d'utilisation habituels de l'API. Les frais supplémentaires sont décrits dans la section AWS IoT Core Tarification - Messagerie.

Les messages MQTT Will publiés lorsqu'un appareil se déconnecte de façon inattendue entraînent des frais de messagerie décrits dans la section AWS IoT Core Tarification- Messagerie.

Pour plus d'informations sur les coûts de messagerie, consultez la section AWS IoT Core Tarification - Messagerie.

Comparaison des messages MQTT retenus et des sessions permanentes MQTT

Les messages retenus et les sessions permanentes sont des fonctionnalités standard de MQTT qui permettent aux appareils de recevoir des messages publiés alors qu'ils étaient hors ligne. Les messages retenus peuvent être publiés à partir de sessions permanentes. Cette section décrit les principaux aspects de ces fonctionnalités et explique comment elles fonctionnent ensemble.

	Messages retenus	Sessions persistantes
Fonctions principales	Les messages retenus peuvent être utilisés pour configurer ou notifier de grands groupes d'appareils après leur connexion. Les messages retenus peuvent également être utilisés lorsque vous souhaitez que les appareils ne reçoivent que le dernier message publié dans une rubrique après une reconnexion.	Les sessions permanentes sont utiles pour les appareils dotés d'une connectivité intermittente et susceptib les de manquer plusieurs messages importants. Les appareils peuvent se connecter à une session permanente pour recevoir les messages envoyés lorsqu'ils sont hors ligne.

	Messages retenus	Sessions persistantes
Exemples	Les messages retenus peuvent fournir aux appareils des informations de configuration relatives à leur environnement lorsqu'ils se connectent. La configuration initiale peut inclure une liste d'autres rubriques de message auxquels il doit s'abonner ou des informations sur la façon dont il doit configurer son fuseau horaire local.	Les appareils qui se connectent via un réseau cellulaire à connectivité intermittente peuvent utiliser des sessions permanentes pour éviter de manquer des messages importants envoyés alors qu'un appareil n'est pas couvert par le réseau ou doit éteindre sa radio cellulaire.
Messages reçus lors de l'inscription initiale à une rubrique	Une fois que vous vous êtes abonné à une rubrique contenant un message retenu, le message retenu le plus récent est reçu.	Une fois que vous vous êtes abonné à une rubrique sans qu'un message soit retenu, aucun message n'est reçu tant qu'un message n'est pas publié dans la rubrique.
Rubriques souscrites après reconnexion	En l'absence de session permanente, le client doit s'abonner aux rubriques après s'être reconnecté.	Les rubriques auxquelles vous êtes abonné sont restaurées après la reconnexion.
Messages reçus après la reconnexion	Une fois que vous vous êtes abonné à une rubrique contenant un message retenu, le message retenu le plus récent est reçu.	Tous les messages publiés avec un QOS = 1 et auxquels vous êtes abonné avec un QOS =1 alors que l'appareil était déconnecté sont envoyés après la reconnexion de l'appareil.

	Messages retenus	Sessions persistantes
Expiration de session/données	Dans MQTT 3, les messages retenus n'expirent pas. Ils sont stockés jusqu'à ce qu'ils soient remplacés ou supprimés. Dans MQTT 5, les messages retenus expirent après l'interva lle d'expiration des messages que vous avez défini. Pour plus d'informations, consultez Expiration de messages.	Les sessions permanentes expirent si le client ne se reconnecte pas dans le délai imparti. Après l'expiration d'une session permanente, les abonnements du client et les messages enregistrés publiés avec un QOS = 1 et auxquels vous avez souscrit avec un QOS =1 alors que l'appareil était déconnect é sont supprimés. Les messages expirés ne seront pas remis. Pour de amples informations sur l'expiration des sessions permanentes, veuillez consulter the section called "Sessions permanentes MQTT".

Pour plus d'informations sur les sessions permanentes, consultez the section called "Sessions permanentes MQTT".

Avec les messages retenus, le client de publication détermine si un message doit être retenu et remis à un appareil après sa connexion, qu'il ait déjà eu une session ou non. Le choix de stocker un message est fait par le diffuseur de publication et le message stocké est remis à tous les clients actuels et futurs qui s'abonnent avec un abonnement QoS 0 ou QoS 1. Les messages retenus ne contiennent qu'un seul message à la fois sur une rubrique donnée.

Lorsqu'un compte a stocké le nombre maximum de messages retenus, AWS IoT Core renvoie une réponse limitée aux messages publiés avec l'option RETAIN définie et à des charges utiles supérieures à 0 octet jusqu'à ce que certains messages retenus soient supprimés et que le nombre de messages retenus tombe en dessous de la limite.

MQTT a conservé les messages et AWS IoT Device Shadows

Les messages retenus et les Device Shadows retiennent les données d'un appareil, mais ils se comportent différemment et répondent à des objectifs différents. Cette section décrit leurs similitudes et leurs différences.

	Messages retenus	Device Shadows
La charge utile des messages possède une structure ou un schéma prédéfini	Tel que défini par l'impléme ntation. MQTT ne spécifie pas de structure ou de schéma pour la charge utile de ses messages.	AWS IoT prend en charge une structure de données spécifiqu e.
La mise à jour de la charge utile des messages génère des messages d'événement	La publication d'un message retenu envoie le message aux clients abonnés, mais ne génère pas de messages de mise à jour supplémentaires.	La mise à jour d'un Device Shadow génère des messages de mise à jour décrivant le changement.
Les mises à jour des messages sont numérotées	Les messages retenus ne sont pas numérotés automatiq uement.	Les documents Device Shadow sont dotés de numéros de version et d'horodatage automatiques.
La charge utile du message est attachée à une ressource d'objet	Les messages retenus ne sont pas attachés à une ressource d'objet.	Les Device Shadows sont attachés à une ressource d'objets.
Mise à jour des éléments individuels de la charge utile du message	Les éléments individuels du message ne peuvent pas être modifiés sans mettre à jour l'intégralité de la charge utile du message.	Les éléments individuels d'un document Device Shadow peuvent être mis à jour sans qu'il soit nécessaire de mettre à jour l'intégralité du document Device Shadow.
Le client reçoit les données des messages lors de l'abonnement	Le client reçoit automatiq uement un message retenu après s'être abonné à une	Les clients peuvent s'abonner aux mises à jour de Device Shadow, mais ils doivent

	Messages retenus	Device Shadows
	rubrique contenant un message retenu.	demander délibérément l'état actuel.
Indexation et consultabilité	Les messages retenus ne sont pas indexés pour la recherche.	L'indexation de flotte indexe les données Device Shadow à des fins de recherche et d'agrégation.

Messages MQTT Last Will and Testament (LWT)

Last Will and Testament (LWT) est une fonctionnalité de MQTT. Grâce LWT, les clients peuvent spécifier un message que l'agent publiera sur un rubrique défini par le client et enverra à tous les clients abonnés au rubrique en cas de déconnexion non initiée. Le message spécifié par les clients est appelé message LWT ou message Will, et la rubrique définie par les clients est appelé rubrique Will. Vous pouvez spécifier un message LWT lorsqu'un appareil se connecte à l'agent. Ces messages peuvent être retenus en plaçant l'indicateur `Will Retain` dans le champ `Connect Flag bits` pendant la connexion. Par exemple, si l'indicateur `Will Retain` est défini sur `1`, un message Will sera stocké dans l'agent dans le rubrique Will associée. Pour plus d'informations, consultez Messages volontaires.

L'agent retiendra les messages Will jusqu'à ce qu'une déconnexion non initiée se produise. Lorsque cela se produit, le courtier publiera les messages à tous les clients abonnés à la rubrique Will pour notifier la déconnexion. Si le client se déconnecte de l'agent avec une déconnexion initiée par le client à l'aide du message MQTT DISCONNECT, l'agent ne publiera pas les messages LWT enregistrés. Dans tous les autres cas, les messages LWT seront envoyés. Pour une liste complète des scénarios de déconnexion dans lesquels l'agent enverra les messages LWT, consultez la section Événements de connexion/déconnexion.

Utilisation de ConnectAttributes

`ConnectAttributes` vous permettent de spécifier les attributs que vous souhaitez utiliser dans votre message de connexion dans vos politiques IAM telles que `PersistentConnect` et `LastWill`. Grâce à `ConnectAttributes`, vous pouvez ainsi créer des politiques qui n'autorisent pas les appareils à accéder aux nouvelles fonctionnalités par défaut, ce qui peut être utile si un appareil est compromis.

`connectAttributes` prend en charge les fonctions suivantes :

`PersistentConnect`

Utilisez la fonctionnalité `PersistentConnect` pour enregistrer tous les abonnements effectués par le client pendant la connexion lorsque la connexion entre le client et le courtier est interrompue.

`LastWill`

Utilisez la fonctionnalité `LastWill` pour publier un message `LastWillTopic` lorsqu'un client se déconnecte de manière inattendue.

Par défaut, votre politique prévoit une connexion non permanente et aucun attribut n'est transmis pour cette connexion. Vous devez spécifier une connexion permanente dans votre politique IAM si vous souhaitez en avoir une.

Pour des exemples `ConnectAttributes`, consultez la section Exemples de politiques de connexion.

Fonctionnalités prises en charge par MQTT 5

AWS IoT Core le support de MQTT 5 est basé sur la spécification MQTT v5.0 avec quelques différences, comme indiqué dans. the section called "AWS IoT différences par rapport aux spécifications MQTT"

AWS IoT Core prend en charge les fonctionnalités MQTT 5 suivantes :

- Abonnements partagés
- Démarrage correct et expiration de session
- Code de raison sur tous ACKs
- Alias de rubrique
- Expiration du message
- Autres fonctionnalités de MQTT 5

Abonnements partagés

AWS IoT Core prend en charge les abonnements partagés pour MQTT 3 et MQTT 5. Les abonnements partagés permettent à plusieurs clients de partager un abonnement à une rubrique et un seul client recevra les messages publiés sur cette rubrique selon une distribution aléatoire. Les abonnements partagés peuvent équilibrer efficacement la charge des messages MQTT entre

un certain nombre d'abonnés. Supposons, par exemple, que 1000 appareils publient sur le même rubrique et que 10 applications principales traitent ces messages. Dans ce cas, les applications principales peuvent s'abonner à la même rubrique et chacune recevra aléatoirement des messages publiés par les appareils sur le rubrique partagé. Cela revient à « partager » efficacement la charge de ces messages. Les abonnements partagés permettent également une meilleure résilience. Lorsqu'une application principale se déconnecte, l'agent répartit la charge entre les abonnés restants du groupe.

Pour utiliser les abonnements partagés, les clients s'abonnent au filtre de rubrique d'un abonnement partagé comme suit :

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

- `$share` est une chaîne littérale indiquant le filtre de rubrique d'un abonnement partagé, qui doit commencer par `$share`.
- `{ShareName}` est une chaîne de caractères qui indique le nom partagé utilisé par un groupe d'abonnés. Le filtre de rubrique d'un abonnement partagé doit contenir un `ShareName` et être suivi du caractère `/`. Le `{ShareName}` ne doit pas inclure les caractères suivants : `/`, `+`, ou`#`. La taille maximale de `{ShareName}` est de 128 octets.
- `{TopicFilter}` suit la même syntaxe de filtre de rubrique qu'un abonnement non partagé. La taille maximale de `{TopicFilter}` est de 256 octets.
- Les deux barres obliques requises (`/`) car les `$share/{ShareName}/{TopicFilter}` ne sont pas incluses dans le nombre maximum de barres obliques dans la rubrique et dans la limite du filtre de rubrique.

Les abonnements identiques `{ShareName}/{TopicFilter}` appartiennent au même groupe d'abonnements partagés. Vous pouvez créer plusieurs groupes d'abonnements partagés et ne pas dépasser la limite d'abonnements partagés par groupe. Pour plus d'informations, veuillez consulter la rubrique Points de terminaison et quotas AWS IoT Core depuis la Référence générale AWS .

Les tableaux suivants comparent les abonnements non partagés et les abonnements partagés :

Abonnement	Description	Exemples de filtres de rubrique
Abonnements non partagés	Chaque client crée un abonnement distinct pour recevoir les messages publiés. Lorsqu'un	`sports/tennis`

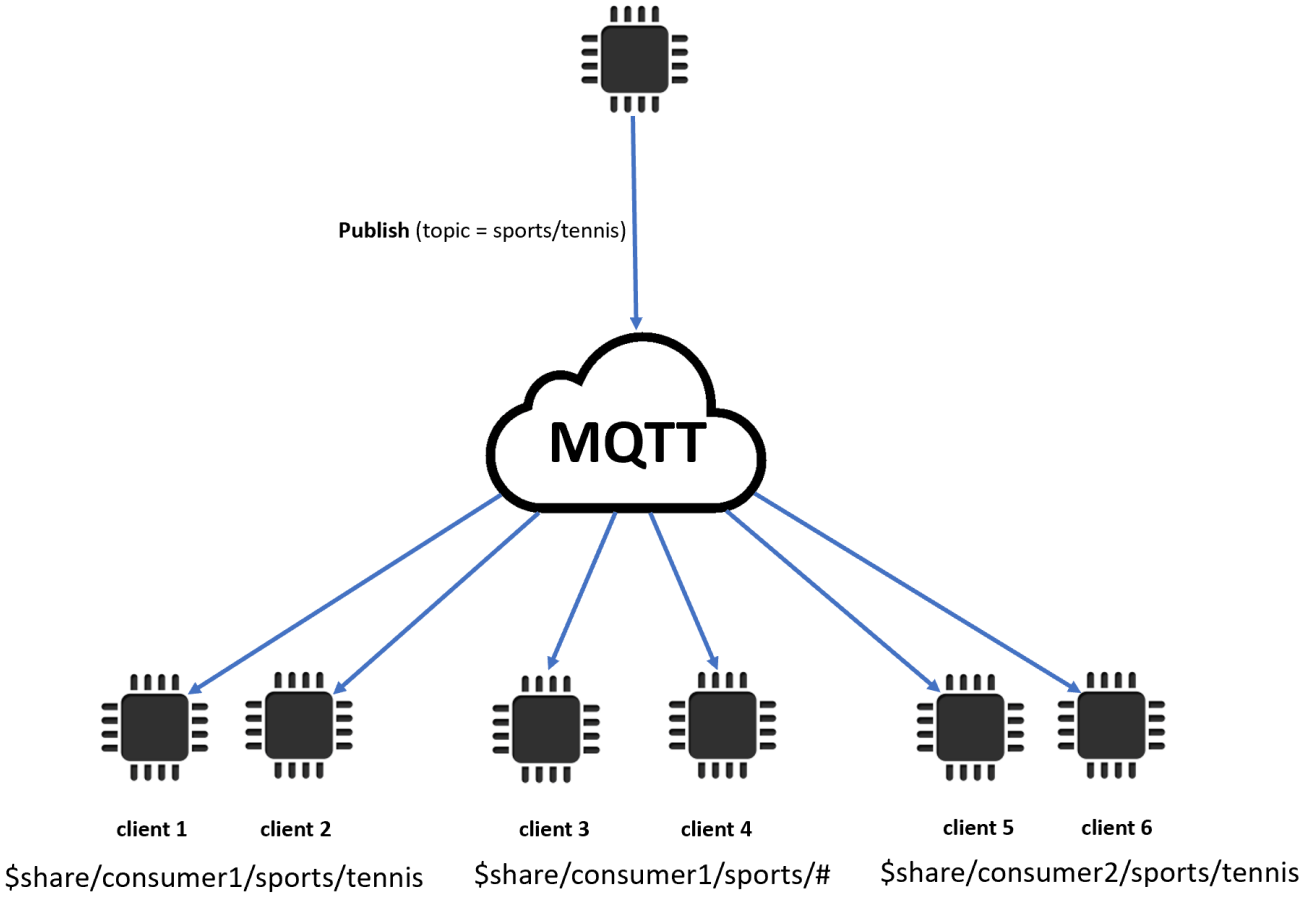
Abonnement	Description	Exemples de filtres de rubrique
	message est publié dans une rubrique, tous les abonnés à ce rubrique reçoivent une copie du message.	`sports/#`
Abonnements partagés	Plusieurs clients peuvent partager un abonnement à un rubrique et un seul client recevra les messages publiés sur ce rubrique de manière aléatoire.	`$share/consumer/sports/tennis` `$share/consumer/sports/#`

Flux d'abonnements non partagés	Flux d'abonnements partagés		

Remarques importantes concernant l'utilisation des abonnements partagés

- Lorsqu'une tentative de publication auprès d'un abonné QoS0 échoue, aucune nouvelle tentative n'a lieu et le message est supprimé.
- Lorsqu'une tentative de publication à destination d'un abonné QoS1 avec une session propre échoue, le message est envoyé à un autre abonné du groupe pour plusieurs tentatives de nouvelle tentative. Les messages qui ne parviennent pas à être remis après toutes les tentatives seront supprimés.
- Lorsqu'une tentative de publication auprès d'un abonné QoS1 avec des sessions permanentes échoue parce que l'abonné est hors ligne, les messages ne sont pas mis en file d'attente et sont envoyés à un autre abonné du groupe. Les messages qui ne parviennent pas à être remis après toutes les tentatives seront supprimés.
- Les abonnements partagés ne reçoivent pas de messages retenus.

- Lorsque les abonnements partagés contiennent des caractères génériques (# ou +), plusieurs abonnements partagés peuvent correspondre à un rubrique. Dans ce cas, l'agent de messages copie le message de publication et l'envoie à un client aléatoire dans chaque abonnement partagé correspondant. Le comportement des caractères génériques des abonnements partagés peut être expliqué dans le schéma suivant.



Dans cet exemple, trois abonnements partagés correspondent à la rubrique MQTT de publication `sports/tennis`. L'agent de messages copie le message publié et l'envoie à un client aléatoire dans chaque groupe correspondant.

Le client 1 et le client 2 partagent l'abonnement : `$share/consumer1/sports/tennis`

Le client 3 et le client 4 partagent l'abonnement : `$share/consumer1/sports/#`

Le client 5 et le client 6 partagent l'abonnement : `$share/consumer2/sports/tennis`

Pour plus d'informations sur les limites des abonnements partagés, consultez la section AWS IoT Core Points de terminaison et quotas de la AWS référence générale. Pour tester les abonnements partagés à l'aide du client AWS IoT MQTT dans la AWS IoT console, consultez???. Pour plus d'informations sur les abonnements partagés, consultez la section Abonnements partagés de la spécification MQTTv5 .0.

Démarrage correct et expiration de session

Vous pouvez utiliser Clean Start et Session Expiration pour gérer vos sessions permanentes avec plus de flexibilité. Un indicateur Clean Start indique si la session doit démarrer sans utiliser une session existante. Un intervalle d'expiration de session indique la durée pendant laquelle la session doit être retenue après une déconnexion. L'intervalle d'expiration des sessions peut être modifié lors de la déconnexion. Pour de plus amples informations, veuillez consulter the section called "Sessions permanentes MQTT".

Code de raison sur tous ACKs

Vous pouvez déboguer ou traiter les messages d'erreur plus facilement à l'aide des codes de motif. Les codes de motif sont renvoyés par l'agent de messages en fonction du type d'interaction avec l'agent (s'abonner, publier, accuser réception). Pour plus d'informations, consultez Codes de motif MQTT. Pour une liste complète des codes de motif MQTT, consultez la spécification MQTT v5.

Alias de rubrique

Vous pouvez remplacer le nom d'une rubrique par un alias de rubrique, qui est un entier de deux octets. L'utilisation d'alias de rubrique permet d'optimiser la transmission des noms de rubriques afin de réduire potentiellement les coûts de données sur les services de données mesurés. AWS IoT Core a une limite par défaut de 8 alias de rubrique. Pour plus d'informations, veuillez consulter la rubrique Points de terminaison et quotas AWS IoT Core depuis la Référence générale AWS .

Expiration du message

Vous pouvez ajouter des valeurs d'expiration aux messages publiés. Ces valeurs représentent l'intervalle d'expiration des messages en secondes. Si un message n'a pas été envoyé aux abonnés dans cet intervalle, il expirera et sera supprimé. Si vous ne définissez pas la valeur d'expiration du message, celui-ci n'expirera pas.

À l'aller, l'abonné recevra un message indiquant le temps restant dans l'intervalle d'expiration. Par exemple, si un message de publication entrant expire 30 secondes et qu'il est acheminé vers l'abonné au bout de 20 secondes, le champ d'expiration du message sera mis à jour à 10. Il est

possible que le message reçu par l'abonné ait un MEI mis à jour de 0. En effet, dès que le temps restant est inférieur ou égal à 999 ms, il sera mis à jour à 0.

Dans AWS IoT Core, l'intervalle d'expiration minimal des messages est de 1. Si l'intervalle est défini sur 0 du côté client, il sera ajusté à 1. L'intervalle d'expiration maximal des messages est de 604 800 (7 jours). Toute valeur supérieure à cette valeur sera ajustée à la valeur maximale.

Dans la communication entre versions, le comportement d'expiration du message est déterminé par la version MQTT du message de publication entrant. Par exemple, un message expirant envoyé par une session connectée via MQTT5 peut expirer pour les appareils abonnés à des MQTT3 sessions. Le tableau ci-dessous indique comment l'expiration des messages prend en charge les types de messages de publication suivants :

Publier un type de message	Intervalle d'expiration des messages
Publier régulièrement	Si un serveur ne parvient pas à délivrer le message dans le délai spécifié, le message expiré sera supprimé et l'abonné ne le recevra pas. Cela inclut les situations telles que lorsqu'un appareil ne publie pas ses messages QoS 1.
Conserver	Si un message retenu expire et qu'un nouveau client s'abonne à la rubrique, le client ne recevra pas le message lors de son inscription.
Dernier testament	L'intervalle entre les derniers messages testamentaires commence une fois que le client se déconnecte et que le serveur tente de transmettre le dernier testament à ses abonnés.
Messages mis en file d'attente	Si un QoS1 sortant avec intervalle d'expiration des messages expire lorsqu'un client est hors ligne, le client ne recevra pas le message expiré après la reprise de la session permanente.

Autres fonctionnalités de MQTT 5

Déconnexion du serveur

Lorsqu'une déconnexion se produit, le serveur peut envoyer au client de manière proactive un message DISCONNECT pour notifier la fermeture de la connexion avec un code de motif de déconnexion.

Requête/réponse

Les éditeurs peuvent demander qu'une réponse soit envoyée par le destinataire à un rubrique spécifié par le diffuseur de publication lors de la réception.

Taille maximale du paquet

Le client et le serveur peuvent spécifier indépendamment la taille de paquet maximale qu'ils prennent en charge.

Format de charge utile et type de contenu

Vous pouvez spécifier le format de charge utile (binaire, texte) et le type de contenu lorsqu'un message est publié. Ils sont transmis au destinataire du message.

Propriétés MQTT 5

Les propriétés MQTT 5 sont des ajouts importants à la norme MQTT pour prendre en charge les nouvelles fonctionnalités de MQTT 5 telles que l'expiration de session et le modèle de requête/réponse. Dans AWS IoT Core, vous pouvez créer des règles qui peuvent transférer les propriétés des messages sortants ou utiliser HTTP Publish pour publier des messages MQTT avec certaines des nouvelles propriétés.

Le tableau suivant répertorie toutes les propriétés MQTT 5 prises AWS IoT Core en charge.

Propriété	Description	Type d'entré	Paquets
Indicateur de format de charge utile	Valeur booléenne indiquant si la charge utile est formatée en UTF-8.	Octet	PUBLIER, CONNECTER
Type de contenu	Chaîne UTF-8 qui décrit le contenu de la charge utile.	Chaîn UTF-8	PUBLIER, CONNECTER
Rubrique de réponse	Chaîne UTF-8 qui décrit la rubrique dans laquelle le récepteur doit effectuer la publicati on dans le cadre du flux demande-réponse. La rubrique ne doit pas avoir de caractères génériques.	Chaîn UTF-8	PUBLIER, CONNECTER

Propriété	Description	Type d'entré	Paquets
Données de corrélation	Données binaires utilisées par l'expéditeur du message de demande pour identifier la demande à laquelle correspond le message de réponse.	Binaire	PUBLIER, CONNECTER
Propriétés de l'utilisateur	Une paire de chaînes UTF-8. Cette propriété peut apparaître plusieurs fois dans un même paquet. Les destinataires recevront les paires clé-valeur dans l'ordre dans lequel elles sont envoyées.	Paire de chaîne UTF-8	CONNECTER, PUBLIER, Will Properties, S'ABONNER, SE DÉCONNECTER, SE DÉSABONNER
Intervalle d'expiration des messages	Entier de 4 octets qui représente l'intervalle d'expiration des messages en secondes. S'il est absent, le message n'expire jamais.	Entier de 4 octet	PUBLIER, CONNECTER
Intervalle d'expiration des sessions	Un entier de 4 octets qui représente l'intervalle d'expiration de la session en secondes. AWS IoT Core prend en charge un maximum de 7 jours, avec un maximum d'une heure par défaut. Si la valeur que vous avez définie dépasse le maximum de votre compte, la valeur ajustée AWS IoT Core sera renvoyée dans le CONNACK.	Entier de 4 octet	CONNECTER , CONNECTER, DÉCONNECTER
Identifiant client attribué	Un identifiant client aléatoire généré AWS IoT Core lorsqu'aucun identifiant client n'est spécifié par les appareils. L'identifiant client aléatoire doit être un nouvel identifiant client qui n'est utilisé par aucune autre session actuellement gérée par le courtier.	Chaîne UTF-8	CONNACK

Propriété	Description	Type d'entré	Paquets
Serveur Keep Alive	Un entier de 2 octets qui représente la durée de rétention attribuée par le serveur. Le serveur déconnecte le client s'il est inactif pendant une durée supérieure à la durée de conservation.	Entier de 2 octe	CONNACK
Demander des informations sur le problème	Valeur booléenne qui indique si la chaîne de motif ou les propriétés utilisateur sont envoyées en cas d'échec.	Octet	CONNECT
Recevoir maximum	Un entier de 2 octets qui représente le nombre maximum de paquets PUBLISH QOS > 0 qui peuvent être envoyés sans recevoir de PUBACK.	Entier de 2 octe	CONNECTER, CONNECTER
Alias maximal du rubrique	Cette valeur indique la valeur la plus élevée qui sera acceptée comme alias de rubrique. La valeur par défaut est 0.	Entier de 2 octe	CONNECTER, CONNECTER
QoS maximale	La valeur maximale de QoS prise en charge. AWS IoT Core La valeur par défaut est 1. AWS IoT Core ne prend pas en charge QoS2.	Octet	CONNACK
Retenir disponible	Valeur booléenne qui indique si le courtier de AWS IoT Core messages prend en charge les messages conservés. La valeur par défaut est 1.	Octet	CONNACK
Taille maximale du paquet	Taille maximale des paquets qui AWS IoT Core acceptent et envoient. Ne peut pas dépasser 128 Ko.	Entier de 4 octe	CONNECTER, CONNECTER

Propriété	Description	Type d'entré	Paquets
Abonnement Wildcard disponible	Valeur booléenne qui indique si le courtier de AWS IoT Core messages prend en charge Wildcard Subscription Available. La valeur par défaut est 1.	Octet	CONNACK
Identifiant d'abonnement disponible	Valeur booléenne qui indique si le courtier de AWS IoT Core messages prend en charge l'identifiant d'abonnement disponible. La valeur par défaut est 0.	Octet	CONNACK

Codes de motif MQTT

MQTT 5 introduit un meilleur signalement des erreurs avec les réponses au code de raison. AWS IoT Core peut renvoyer des codes de motif, y compris, mais sans s'y limiter, les suivants regroupés par paquets. Pour une liste complète des codes de motif pris en charge par MQTT 5, consultez les spécifications du MQTT 5.

Codes de motif CONNACK

Valeur	Hex	Nom du code de motif	Description
0	0x00	Réussite	La connexion est acceptée.
128	0x80	Erreur non spécifiée	Le serveur ne souhaite pas révéler le motif de l'échec, sinon aucun des autres codes de motif ne s'applique.
133	0x85	Identifiant du client non valide	L'identifiant du client est une chaîne valide mais n'est pas autorisé par le serveur.
134	0x86	Nom d'utilisateur ou mot de passe incorrect	Le serveur n'accepte pas le nom d'utilisateur ou le mot de passe spécifiés par le client.
135	0x87	Non autorisé	Le client n'est pas autorisé à se connecter.

Valeur	Hex	Nom du code de motif	Description
144	0x90	Nom de la rubrique non valide	Le nom de la rubrique testamentaire est correctement formé mais n'est pas accepté par le serveur.
151	0x97	Quota dépassé	Une limite de mise en œuvre ou imposée par l'adminis tration a été dépassée.
155	0 x 9 B	QoS n'est pas pris en charge.	Le serveur ne prend pas en charge la QoS définie dans Will QoS.

Codes de motif PUBACK

Valeur	Hex	Nom du code de motif	Description
0	0x00	Réussite	Le message est accepté. La publication du message QoS 1 se poursuit.
128	0x80	Erreur non spécifiée	Le destinataire n'accepte pas la publication, mais soit ne souhaite pas en révéler la raison, soit elle ne correspond pas à l'une des autres valeurs.
135	0x87	Non autorisé	Le PUBLISH n'est pas autorisé.
144	0x90	Nom de la rubrique non valide	Le nom de la rubrique n'est pas mal formé, mais il n'est pas accepté par le client ou le serveur.
145	0x91	Identifiant de paquet en cours d'utilisa tion	L'identifiant du paquet est déjà utilisé. Cela peut indiquer une incompatibilité de l'état de session entre le client et le serveur.
151	0x97	Quota dépassé	Une limite de mise en œuvre ou imposée par l'adminis tration a été dépassée.

Code de motif de déconnexion

Valeur	Hex	Nom du code de motif	Description
129	0x81	Paquet incorrect	Le paquet reçu n’est pas conforme à cette spécification.
130	0x82	Erreur de protocole	Un paquet inattendu ou en panne a été reçu.
135	0x87	Non autorisé	La demande n'est pas autorisée.
139	0 x 8 B	Arrêt du serveur	Le serveur est en train de s’arrêter.
141	0 x 8D	Délai d’expiration de Keep Alive	La connexion est fermée car aucun paquet n’a été reçu pendant 1,5 fois la durée de Keep Alive.
142	0 x 8E	Session prise en charge	Une autre connexion utilisant le même ClientID s’est connectée, ce qui a entraîné la fermeture de cette connexion.
143	0 x 8 F	Filtre de rubrique invalide	Le filtre de rubrique est correctement formé mais n’est pas accepté par le serveur.
144	0x90	Nom de la rubrique non valide	Le nom de la rubrique est correctement formé mais n’est pas accepté par ce client ou serveur.
147	0x93	Dépassement du maximum de réception	Le client ou le serveur a reçu une publication supérieure à la valeur maximale de réception pour laquelle il n’a pas envoyé de PUBACK ou de PUBCOMP.
148	0x94	Alias de rubrique invalide	Le client ou le serveur a reçu un paquet PUBLISH contenant un alias de rubrique supérieur à l’alias de rubrique maximal qu’il a envoyé dans le paquet CONNECT ou CONNACK.

Valeur	Hex	Nom du code de motif	Description
151	0x97	Quota dépassé	Une limite de mise en œuvre ou imposée par l'adminis tration a été dépassée.
152	0x98	Action administrative	La connexion est interrompue suite à une action administr ative.
155	0 x 9 B	QoS n'est pas pris en charge.	Le client a spécifié une QoS supérieure à la QoS spécifiée dans une QoS maximale dans le CONNACK.
161	0 x A1	Identifiants d'abonnem ent non pris en charge	Le serveur ne prend pas en charge les identifiants d'abonnement ; l'abonnement n'est pas accepté.

Codes de motif SUBACK

Valeur	Hex	Nom du code de motif	Description
0	0x00	QoS 0 accordé	L'abonnement est accepté et la QoS maximale envoyée sera de QoS 0. Il s'agit peut-être d'une QoS inférieure à celle demandée.
1	0x01	QoS 1 accordé	L'abonnement est accepté et la QoS maximale envoyée sera de QoS 1. Il s'agit peut-être d'une QoS inférieure à celle demandée.
128	0x80	Erreur non spécifiée	L'abonnement n'est pas accepté et le serveur ne souhaite pas en révéler le motif ou aucun des autres codes de motif ne s'applique.
135	0x87	Non autorisé	Le client n'est pas autorisé à souscrire cet abonnement.
143	0 x 8 F	Filtre de rubrique invalide	Le filtre de rubrique est correctement formé mais n'est pas autorisé pour ce client.

Valeur	Hex	Nom du code de motif	Description
145	0x91	Identifiant de paquet en cours d'utilisation	L'identifiant de paquet spécifié est déjà utilisé.
151	0x97	Quota dépassé	Une limite de mise en œuvre ou imposée par l'administration a été dépassée.

Codes de motif UNSUBACK

Valeur	Hex	Nom du code de motif	Description
0	0x00	Réussite	L'abonnement est supprimé.
128	0x80	Erreur non spécifiée	Le désabonnement n'a pas pu être effectué et le serveur ne souhaite pas en révéler le motif ou aucun des autres codes de motif ne s'applique.
143	0 x 8 F	Filtre de rubrique invalide	Le filtre de rubrique est correctement formé mais n'est pas autorisé pour ce client.
145	0x91	Identifiant de paquet en cours d'utilisation	L'identifiant de paquet spécifié est déjà utilisé.

AWS IoT différences par rapport aux spécifications MQTT

L'implémentation de l'agent de messages est basée sur les spécifications MQTT v3.1.1 et MQTT v5.0, mais elle diffère des spécifications de la manière suivante :

- AWS IoT ne prend pas en charge les paquets suivants pour MQTT 3 : PUBREC, PUBREL et PUBCOMP.
- AWS IoT ne prend pas en charge les paquets suivants pour MQTT 5 : PUBREC, PUBREL, PUBCOMP et AUTH.

- AWS IoT ne prend pas en charge la redirection du serveur MQTT 5.
- AWS IoT prend en charge les niveaux de qualité de service (QoS) MQTT 0 et 1 uniquement. AWS IoT ne prend pas en charge la publication ou l'abonnement avec QoS de niveau 2. Lorsque QoS 2 est requis, l'agent de messages , n'envoie pas de PUBACK ou de SUBACK.
- En effet AWS IoT, l'abonnement à une rubrique dont le niveau de QoS est 0 signifie qu'un message est délivré zéro fois ou plus. Un message peut être remis plusieurs fois. Les messages remis plusieurs fois peuvent être envoyés avec un ID de paquet différent. Dans ce cas, l'indicateur DUP n'est pas défini.
- Lorsqu'il répond à une demande de connexion, l'agent de messages envoie un message CONNACK. Ce message contient un indicateur précisant si la connexion reprend une session précédente.
- Avant d'envoyer des paquets de contrôle supplémentaires ou une demande de déconnexion, le client doit attendre que le message CONNACK soit reçu sur son appareil par l'agent de messages AWS IoT .
- Lorsqu'un client s'abonne à une rubrique, il peut y avoir un délai entre le moment où l'agent de messages envoie un SUBACK et le moment où le client commence à recevoir de nouveaux messages correspondants.
- Lorsqu'un client utilise le caractère générique # dans le filtre de rubrique pour s'abonner à une rubrique, toutes les chaînes situées à son niveau ou inférieur dans la hiérarchie des rubriques sont mises en correspondance. Cependant, la rubrique parent ne correspond pas. Par exemple, un abonnement à la rubrique `sensor/#` reçoit les messages publiés dans les rubriques `sensor/`, `sensor/temperature`, `sensor/temperature/room1`, mais pas les messages publiés dans `sensor`. Pour plus d'informations sur les caractères génériques, consultez Filtres de noms de sujets.
- L'agent de messages utilise l'ID de client pour identifier chaque client. L'ID de client est transmis depuis le client à l'agent de messages dans le cadre de la charge utile MQTT. Deux clients possédant le même ID de client ne peuvent pas être connectés simultanément à l'agent de messages. Lorsqu'un client se connecte à l'agent de messages à l'aide d'un ID de client qu'un autre client utilise, la nouvelle connexion client est acceptée et le client connecté précédemment est déconnecté.
- À de rares occasions, l'agent de messages peut renvoyer le même message PUBLISH logique avec un ID de paquet différent.
- L'abonnement aux filtres de rubrique contenant un caractère générique ne permet pas de recevoir les messages retenus. Pour recevoir un message retenu, la demande d'abonnement doit contenir un filtre de rubrique correspondant exactement à la rubrique du message retenu.

- L’agent de messages ne garantit pas l’ordre dans lequel les messages et les ACK sont reçus.
- AWS IoT peut avoir des limites différentes des spécifications. Pour plus d’informations veuillez consulter AWS IoT Core Limites et quotas de l’agent de messages et des protocoles dans le AWS IoT Guide de référence.
- L'indicateur MQTT DUP n'est pas pris en charge.

HTTPS

Les clients peuvent publier des messages en adressant des demandes à l'API REST à l'aide des protocoles HTTP 1.0 ou 1.1. Pour connaître l'authentification et les mappages de port utilisés par les demandes HTTP, veuillez consulter ???.

> **Note**
>
> HTTPS ne prend pas en charge une valeur `clientId` comme le fait MQTT. `clientId` est disponible lors de l'utilisation de MQTT, mais pas disponible lors de l'utilisation de HTTPS.

URL du message HTTPS

Les appareils et les clients publient leurs messages en envoyant des requêtes POST à un point de terminaison spécifique au client et à une URL spécifique au sujet :

```
https://IoT_data_endpoint/topics/url_encoded_topic_name?qos=1
```

- *`IoT_data_endpoint`*est le point de terminaison des données de l'AWS IoT appareil. Vous pouvez trouver le point de terminaison dans la AWS IoT console, sur la page de détails de l'objet ou sur le client à l'aide de la AWS CLI commande :

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

  Le point de terminaison doit ressembler à ceci : `a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`
- *`url_encoded_topic_name`*est le nom complet du sujet du message envoyé.

Exemples de code de message HTTPS

Voici comment envoyer un message HTTPS à AWS IoT.

Python (port 8443)

```
import requests
import argparse

# define command-line parameters
parser = argparse.ArgumentParser(description="Send messages through an HTTPS
 connection.")
parser.add_argument('--endpoint', required=True, help="Your AWS IoT data custom
 endpoint, not including a port. " +
                                                       "Ex: \"abcdEXAMPLExyz-
ats.iot.us-east-1.amazonaws.com\"")
parser.add_argument('--cert', required=True, help="File path to your client
 certificate, in PEM format.")
parser.add_argument('--key', required=True, help="File path to your private key, in
 PEM format.")
parser.add_argument('--topic', required=True, default="test/topic", help="Topic to
 publish messages to.")
parser.add_argument('--message', default="Hello World!", help="Message to publish. "
 +
                                                       "Specify empty string to
 publish nothing.")

# parse and load command-line parameter values
args = parser.parse_args()

# create and format values for HTTPS request
publish_url = 'https://' + args.endpoint + ':8443/topics/' + args.topic + '?qos=1'
publish_msg = args.message.encode('utf-8')

# make request
publish = requests.request('POST',
            publish_url,
            data=publish_msg,
            cert=[args.cert, args.key])

# print results
print("Response status: ", str(publish.status_code))
if publish.status_code == 200:
```

```
        print("Response body:", publish.text)
```

Python (port 443)

```
import requests
import http.client
import json
import ssl

ssl_context = ssl.SSLContext(protocol=ssl.PROTOCOL_TLS_CLIENT)
ssl_context.minimum_version = ssl.TLSVersion.TLSv1_2

# note the use of ALPN
ssl_context.set_alpn_protocols(["x-amzn-http-ca"])
ssl_context.load_verify_locations(cafile="./<root_certificate>")

# update the certificate and the AWS endpoint
ssl_context.load_cert_chain("./<certificate_in_PEM_Format>",
 "<private_key_in_PEM_format>")
connection = http.client.HTTPSConnection('<the ats IoT endpoint>', 443,
 context=ssl_context)
message = {'data': 'Hello, I'm using TLS Client authentication!'}
json_data = json.dumps(message)
connection.request('POST', '/topics/device%2Fmessage?qos=1', json_data)

# make request
response = connection.getresponse()

# print results
print(response.read().decode())
```

CURL

Vous pouvez utiliser curl à partir d'un client ou d'un appareil pour envoyer un message à AWS IoT.

Pour utiliser curl pour envoyer un message depuis un appareil AWS IoT client

1. Vérifiez la version curl.

   a. Sur votre client, exécutez cette commande à partir d’une invite de commande.

      curl --help

Dans le texte d'aide, recherchez les options TLS. Vous devriez voir l'option `--tlsv1.2`.

b. Si vous voyez l' option `--tlsv1.2`, continuez.

c. Si vous ne voyez pas l'option `--tlsv1.2` ou si vous obtenez une erreur `command not found`, vous devrez peut-être mettre à jour ou installer curl sur votre client ou l'installer `openssl`avant de continuer.

2. Installez les certificats sur votre client.

   Copiez les fichiers de certificat que vous avez créés lorsque vous avez enregistré votre client (objet) dans la AWS IoT console. Assurez-vous d'avoir ces trois fichiers de certificat sur votre client avant de continuer.

   - Fichier de certificat de l'autorité de certification (*`Amazon-root-CA-1.pem`* dans cet exemple).
   - Fichier de certificat du client (*`device.pem.crt`* dans cet exemple).
   - Fichier de clé privée du client (*`private.pem.key`* dans cet exemple).

3. Créez la ligne de commande curl en remplaçant les valeurs remplaçables par celles de votre compte et de votre système.

```
curl --tlsv1.2 \
    --cacert Amazon-root-CA-1.pem \
    --cert device.pem.crt \
    --key private.pem.key \
    --request POST \
    --data "{ \"message\": \"Hello, world\" }" \
    "https://IoT_data_endpoint:8443/topics/topic?qos=1"
```

--tlsv1.2

Utilisez TLS 1.2 (SSL).

--cacert *`Amazon-root-CA-1.pem`*

Nom et chemin d’accès du fichier de certificat d'autorité de certification, si nécessaire, pour vérifier l'appairage.

--certificat *`device.pem.crt`*

Nom et chemin d’accès du fichier de certificat du client, si nécessaire.

--clé *private.pem.key*

Nom et chemin d'accès du fichier de clé privée du client, si nécessaire.

--demande POST

Type de demande HTTP (dans le cas présent, POST).

--données « » *{ \"message\": \"Hello, world\" }*

Données POST HTTP que vous souhaitez publier. Dans ce cas, il s'agit d'une chaîne JSON, avec les guillemets internes échappés à l'aide du caractère de barre oblique inverse (\).

« https : *IoT_data_endpoint* //:8443/sujets/ ? *topic* qos=1 »

L'URL du point de terminaison des données de l' AWS IoT appareil de votre client`:8443`, suivie du port HTTPS, qui est ensuite suivi du mot clé `/topics/` et du nom du sujet`topic`, dans ce cas. Spécifiez la qualité du service en tant que paramètre de requête, `?qos=1`.

4. Ouvrez le client de test MQTT dans la AWS IoT console.

   Suivez les instructions Afficher les messages MQTT avec le client AWS IoT MQTT et configurez la console pour vous abonner aux messages dont le nom de sujet est *topic* utilisé dans votre curl commande, ou utilisez le filtre de sujet générique de#.

5. Testez la commande.

   Lors de la surveillance de la rubrique dans le client de test de la console AWS IoT , accédez à votre client et émettez la ligne de commande curl que vous avez créée à l'étape 3. Vous devriez voir les messages de votre client dans la console.

Rubriques MQTT

Les rubriques MQTT identifient les AWS IoT messages. AWS IoT les clients identifient les messages qu'ils publient en leur attribuant des noms de sujets. Les clients identifient les messages auxquels ils souhaitent s'abonner (réception) en enregistrant un filtre de rubrique avec AWS IoT Core. L'agent de messages utilise des noms de rubrique et des filtres de rubrique pour acheminer les messages des clients publiant vers les clients abonnés.

L'agent de messages utilise des rubriques pour identifier les messages envoyés via MQTT et envoyés via HTTP auURL du message HTTPS.

Bien que certains sujets réservés au système soient pris AWS IoT en charge, la plupart des sujets MQTT sont créés et gérés par vous, le concepteur du système. AWS IoT utilise des rubriques pour identifier les messages reçus des clients de publication et sélectionner les messages à envoyer aux clients abonnés, comme décrit dans les sections suivantes. Avant de créer un espace de nom de rubrique pour votre système, passez en revue les caractéristiques des rubriques MQTT pour créer la hiérarchie des noms de rubrique qui fonctionne le mieux pour votre système IoT.

Noms de rubrique

Les noms de rubrique et les filtres de rubrique sont des chaînes codées en UTF-8. Ils peuvent représenter une hiérarchie d'informations en utilisant la barre oblique (/) pour séparer les niveaux de la hiérarchie. Par exemple, ce nom de rubrique peut faire référence à un capteur de température dans la salle 1 :

- `sensor/temperature/room1`

Dans cet exemple, il peut également y avoir d'autres types de capteur dans d'autres pièces avec des noms de rubrique tels que :

- `sensor/temperature/room2`
- `sensor/humidity/room1`
- `sensor/humidity/room2`

> **Note**
>
> Lorsque vous réfléchissez aux noms de rubrique pour les messages de votre système, gardez à l'esprit les points suivants :
>
> - Les noms de rubrique et les filtres de rubrique sont sensibles à la casse.
> - Les noms de rubrique ne doivent pas contenir d'informations personnelles identifiables.
> - Les noms de rubrique commençant par $ sont des rubriques réservées qui ne doivent être utilisées que par AWS IoT Core.

- AWS IoT Core Impossible d'envoyer ou de recevoir des messages entre Compte AWS les régions ou entre elles.

Pour plus d'informations sur la conception des noms de rubriques et de l'espace de noms, consultez notre livre blanc, Conception de rubriques MQTT pour. AWS IoT Core

Pour obtenir des exemples de la façon dont les applications peuvent publier des messages et s'y abonner, commencez par Commencer à utiliser les AWS IoT Core didacticiels etAWS IoT Appareil SDKs, appareil mobile SDKs et client AWS IoT d'appareil.

> **Important**
>
> L'espace de noms de rubrique est limité à une région Compte AWS et. Par exemple, le `sensor/temp/room1` sujet utilisé par un utilisateur Compte AWS dans une région est distinct du `sensor/temp/room1` sujet utilisé par le même AWS compte dans une autre région ou utilisé par un autre Compte AWS dans n'importe quelle région.

ARN de la rubrique

Tous les sujets ARNs (Amazon Resource Names) ont la forme suivante :

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Par exemple, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/application/topic/device/sensor` est un ARN pour la rubrique `application/topic/device/sensor`.

Filtres de noms de sujets

Les clients abonnés enregistrent des filtres de noms de sujets auprès du courtier de messages afin de spécifier les sujets de message que le courtier de messages doit leur envoyer. Un filtre de nom de rubrique peut être un nom de rubrique unique pour s'abonner à un seul nom de rubrique ou il peut inclure des caractères génériques pour s'abonner à plusieurs noms de rubrique en même temps.

Les clients publiant ne peuvent pas utiliser de caractères génériques dans les noms de rubrique qu'ils publient.

Le tableau suivant répertorie les caractères génériques pouvant être utilisés dans un filtre de rubrique.

Caractères génériques de rubrique

Caractère générique	Correspondance	Remarques
#	Toutes les chaînes au niveau et au-dessous dans la hiérarchie des rubriques.	Doit être le dernier caractère du filtre de rubrique. Doit être le seul caractère dans son niveau de hiérarchie des rubriques. Peut être utilisé dans un filtre de rubrique contenant également le caractère générique +.
+	Toute chaîne du niveau qui contient le caractère.	Doit être le seul caractère dans son niveau de hiérarchie des rubriques. Peut être utilisé dans plusieurs niveaux d'un filtre de rubrique.

Utilisation de caractères génériques avec les exemples de nom de rubrique de capteur précédents :

- Un abonnement à `sensor/#` reçoit les messages publiés dans `sensor/`, `sensor/temperature`, `sensor/temperature/room1`, mais pas les messages publiés dans `sensor`.
- Un abonnement à `sensor/+/room1` reçoit les messages publiés dans `sensor/temperature/room1` et `sensor/humidity/room1`, mais pas les messages envoyés à `sensor/temperature/room2` ou `sensor/humidity/room2`.

ARN de filtre de rubriques :

Tous les filtres de rubrique ARNs (Amazon Resource Names) ont la forme suivante :

```
arn:aws:iot:aws-region:AWS-account-ID:topicfilter/TopicFilter
```

Par exemple, `arn:aws:iot:us-west-2:123EXAMPLE456:topicfilter/application/topic/+/sensor` est un ARN pour le filtre de rubrique `application/topic/+/sensor`.

Charge utile du message MQTT

La charge utile des messages envoyés dans vos messages MQTT n'est pas spécifiée par AWS IoT, sauf si c'est pour l'un des. the section called “Rubriques réservées” Pour répondre aux besoins de votre application, nous vous recommandons de définir la charge utile des messages pour vos sujets dans le respect des contraintes du AWS IoT Core Service Quotas pour les protocoles.

L'utilisation d'un format JSON pour la charge utile de vos messages permet au moteur de AWS IoT règles d'analyser vos messages et de leur appliquer des requêtes SQL. Si votre application n'a pas besoin du moteur de règles pour appliquer des requêtes SQL aux charges utiles de vos messages, vous pouvez utiliser le format de données requis par votre application. Pour plus d'informations sur les limitations et les caractères réservés dans un document JSON utilisé dans les requêtes SQL, consultezExtensions JSON.

Pour plus d'informations sur la conception de vos rubriques MQTT et de leurs charges utiles de messages correspondantes, consultez la section Conception de rubriques MQTT pour AWS IoT Core.

Si la limite de taille d'un message dépasse les quotas de service, cela se traduira par un « `CLIENT_ERROR` avec motif » `PAYLOAD_LIMIT_EXCEEDED` et « La charge utile du message dépasse la taille limite pour le type de message ». Pour plus d'informations sur la limite de taille des messages, consultez la section Limites et quotas du courtier de AWS IoT Core messages.

Rubriques réservées

Les rubriques commençant par le signe du dollar ($) sont réservées à l'usage de AWS IoT. Vous pouvez vous abonner à ces rubriques réservées et y publier lorsqu'elles le permettent. Toutefois, vous ne pouvez pas créer de nouvelles rubriques commençant par un signe dollar. Les opérations de publication ou d'abonnement à des rubriques réservées qui ne sont pas prises en charge peuvent entraîner la fin de la connexion.

Rubriques de modèle de ressource

Rubrique	Opérations autorisées du client	Description
$ aws/sitewise/asset - models/ /assets/ /properties/	S’abonner	AWS IoT SiteWise publie des notifications relatives

Rubrique	Opérations autorisées du client	Description
assetModelId assetId propertyId		aux propriétés des actifs dans cette rubrique. Pour plus d'informations, consultez la section Interacti on avec d'autres AWS services dans le guide de AWS IoT SiteWise l'utilisa teur.

AWS IoT Device Defender sujets

Ces messages prennent en charge les tampons de réponse au format CBOR (Concise Binary JavaScript Object Representation) et au format JSON (Object Notation), selon le *payload-format* sujet. AWS IoT Device Defender les rubriques ne supportent que la publication MQTT.

payload-format	Type de données du format de réponse
CBOR	CBOR (Concise Binary Object Representation, représentation concise d'objets binaires)
json	JavaScript Notation d'objets (JSON)

Pour plus d'informations, consultez la section Envoi de métriques depuis des appareils.

Rubrique	Opérations autorisée s	Description
$aws/things/ /defender /metrics/ *thingName payload-format*	Publication	AWS IoT Device Defender les agents publient des métriques sur cette rubrique. Pour plus d'informations, consultez la section Envoi de métriques depuis des appareils.
$aws/things/ /defender /metrics/ /accepted	S’abonner	AWS IoT publie sur cette rubrique après qu'un AWS IoT Device Defender

Rubrique	Opérations autorisées	Description
thingName *payload-format*		agent a publié un message réussi dans *thingName* $aws/things/ /defender/ metrics/. *payload-format* Pour plus d'informations, consultez la section Envoi de métriques depuis des appareils.
$aws/things/ /defender/ metrics/ /rejeté *thingName* *payload-format*	S’abonner	AWS IoT publie sur cette rubrique après qu'un AWS IoT Device Defender agent a publié un message infructueux dans *thingName* $aws/things/ /defender/ metrics/. *payload-format* Pour plus d'informations, consultez la section Envoi de métriques depuis des appareils.

AWS IoT Core Rubriques sur la localisation des appareils

AWS IoT Core La localisation de l'appareil peut résoudre les données de mesure de votre appareil et fournir une estimation de l'emplacement de vos appareils IoT. Les données de mesure de l'appareil peuvent inclure le GNSS, le Wi-Fi, le réseau cellulaire et l'adresse IP. AWS IoT Core Device Location choisit ensuite le type de mesure offrant la meilleure précision et résout les informations de localisation de l'appareil. Pour plus d’informations, consultez AWS IoT Core Emplacement de l'appareil et Résolution de la localisation de l'appareil à l'aide des rubriques MQTT AWS IoT Core relatives à la localisation.

Rubrique	Opérations autorisées	Description
$aws/device_location/ / get_position_estimate *customer_device_id*	Publication	Un appareil publie dans cette rubrique pour obtenir les données de mesure brutes numérisées à résoudre en fonction de l'emplacement de AWS IoT Core l'appareil.
$aws/device_location/ / get_position_estimate/	S’abonner	AWS IoT Core L'emplacement de l'appareil est publié dans cette rubrique

Rubrique	Opérations autorisées	Description
accepted *customer_device_id*		une fois que la localisation de l'appareil a été correctement résolue.
$aws/device_location/ / get_position_estimate/rejeté *customer_device_id*	S'abonner	AWS IoT Core Device Location publie dans cette rubrique lorsqu'il n'est pas possible de résoudre correctement la localisation de l'appareil en raison d'erreurs 4xx.

Rubriques d'événement

Les messages d'événements sont publiés lorsque certains événements se produisent. Par exemple, le registre génère des événements quand des objets sont ajoutés, mis à jour ou supprimés. Le tableau présente les différents AWS IoT événements et les sujets qui leur sont réservés.

Rubrique	Opérations autorisées du client	Description
$aws/events/certificates/registered/*caCertificateId*	S'abonner	AWS IoT publie ce message lors de l'enregistrement AWS IoT automatique d'un certificat et lorsqu'un client présente un certificat avec le `PENDING_ACTIVATION` statut. Pour de plus amples informations, veuillez consulter the section called "Configuration de la première connexion par un client pour l'enregistrement automatique".
$aws/events/job/*jobID*/annulé	S'abonner	AWS IoT publie ce message lorsqu'une tâche est annulée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobID*/annulation_en cours	S'abonner	AWS IoT publie ce message lorsqu'une annulation d'offre d'emploi est en

Rubrique	Opérations autorisées du client	Description
		cours. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobID*/terminé	S'abonner	AWS IoT publie ce message lorsqu'une tâche est terminée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobID*/supprimé	S'abonner	AWS IoT publie ce message lorsqu'une tâche est supprimée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobID*/deletion_in_en cours	S'abonner	AWS IoT publie ce message lorsqu'une tâche est en cours de suppression. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobID*/annulé	S'abonner	AWS IoT publie ce message lorsqu'une exécution de tâche est annulée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobID*/supprimé	S'abonner	AWS IoT publie ce message lorsqu'une exécution de tâche est supprimée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobID*/échec	S'abonner	AWS IoT publie ce message en cas d'échec de l'exécution d'une tâche. Pour de plus amples informations, veuillez consulter Événements Jobs.

Rubrique	Opérations autorisées du client	Description
$aws/events/jobExecution/*jobID*/rejeté	S'abonner	AWS IoT publie ce message lorsqu'une exécution de tâche a été rejetée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobID*/supprimé	S'abonner	AWS IoT publie ce message lorsqu'une exécution de tâche a été supprimée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobID*/réussi	S'abonner	AWS IoT publie ce message lorsqu'une tâche a été exécutée avec succès. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobID*/timed_out	S'abonner	AWS IoT publie ce message lorsque le délai d'exécution d'une tâche a expiré. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/presence/connected/*clientId*	S'abonner	AWS IoT publie sur cette rubrique lorsqu'un client MQTT avec l'ID client spécifié se connecte à AWS IoT. Pour de plus amples informations, veuillez consulter Événements de connexion/déconnexion.
$aws/events/presence/disconnected/*clientId*	S'abonner	AWS IoT publie sur cette rubrique lorsqu'un client MQTT avec l'ID client spécifié se déconnecte de. AWS IoT Pour de plus amples informations, veuillez consulter Événements de connexion/déconnexion.

Rubrique	Opérations autorisées du client	Description
$aws/events/subscriptions/subscribed/*clientId*	S'abonner	AWS IoT publie dans cette rubrique lorsqu'un client MQTT avec l'ID client spécifié s'abonne à une rubrique MQTT. Pour de plus amples informations, veuillez consulter Événements d'abonnement/désabonnement.
$aws/events/subscriptions/unsubscribed/*clientId*	S'abonner	AWS IoT publie dans cette rubrique lorsqu'un client MQTT avec l'ID client spécifié se désabonne d'une rubrique MQTT. Pour de plus amples informations, veuillez consulter Événements d'abonnement/désabonnement.
$aws/events/thing/*thingName* /créé	S'abonner	AWS IoT publie sur cette rubrique lorsque l'*thingName* objet est créé. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thing/*thingName* /mis à jour	S'abonner	AWS IoT publie sur cette rubrique lorsque l'*thingName* objet est mis à jour. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thing/*thingName* /supprimé	S'abonner	AWS IoT publie sur cette rubrique lorsque l'*thingName* objet est supprimé. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".

Rubrique	Opérations autorisées du client	Description
$aws/events/thingGroup/*thingGroupName* /créé	S'abonner	AWS IoT publie sur cette rubrique lors de la création d'un groupe *thingGroupName* d'objets. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingGroup/*thingGroupName* /mis à jour	S'abonner	AWS IoT publie sur cette rubrique lorsque le groupe d'objets *thingGroupName* est mis à jour. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingGroup/*thingGroupName* /supprimé	S'abonner	AWS IoT publie sur cette rubrique lorsque le groupe d'objets *thingGroupName* est supprimé. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingType/*thingTypeName* /créé	S'abonner	AWS IoT publie sur cette rubrique lorsque le type d'*thingTypeName* objet est créé. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingType/*thingTypeName* /mis à jour	S'abonner	AWS IoT publie dans cette rubrique lorsque le type d'*thingTypeName* objet est mis à jour. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingType/*thingTypeName* /supprimé	S'abonner	AWS IoT publie dans cette rubrique lorsque le type d'*thingTypeName* objet est supprimé. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".

Rubrique	Opérations autorisées du client	Description
$aws/events/thingTypeAssociation/thing/*thingName* /*thingTypeName*	S'abonner	AWS IoT publie dans cette rubrique lorsqu'un objet *thingName* est associé ou dissocié d'un type *thingTypeName* d'objet. Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingGroupMembership/thingGroup/*thingGroupName* /thing/ /ajouté *thingName*	S'abonner	AWS IoT publie dans cette rubrique lorsqu'un objet *thingName* est ajouté à un groupe d'objets*thingGroupName* . Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingGroupMembership/thingGroup/*thingGroupName* /thing/ /supprimé *thingName*	S'abonner	AWS IoT publie dans cette rubrique lorsqu'un objet *thingName* est retiré du groupe d'objets*thingGroupName* . Pour de plus amples informations, veuillez consulter the section called "Événements de registre".
$aws/events/thingGroupHierarchy/thingGroup//*parentThingGroupName* childThingGroup/*childThingGroupName* /ajouté	S'abonner	AWS IoT publie sur cette rubrique lorsqu'un groupe d'objets *childThingGroupName* est ajouté à un groupe d'objets*parentThingGroupName* . Pour de plus amples informations, veuillez consulter the section called "Événements de registre".

Rubrique	Opérations autorisées du client	Description
$aws/events/thingGroupHierarchy/thingGroup//*parentThingGroupName* childThingGroup/*childThingGroupName* /supprimé	S'abonner	AWS IoT publie sur cette rubrique lorsque le groupe d'objets *childThingGroupName* est supprimé du groupe d'objets*parentThingGroupName* . Pour de plus amples informations, veuillez consulter the section called "Événements de registre".

Rubriques d'approvisionnement de la flotte

> **Note**
>
> Les opérations client notées « Recevoir » dans ce tableau indiquent les sujets AWS IoT qui sont publiés directement auprès du client qui les a demandés, que le client soit abonné au sujet ou non. Les clients doivent s'attendre à recevoir ces messages de réponse même s'ils ne s'y sont pas abonnés. Ces messages de réponse ne passent pas par le courtier de messages et ne peuvent pas être souscrits par d'autres clients ou règles.

Ces messages prennent en charge les tampons de réponse au format CBOR (Concise Binary JavaScript Object Representation) et au format JSON (Object Notation), selon le *payload-format* sujet.

payload-format	Type de données du format de réponse
CBOR	CBOR (Concise Binary Object Representation, représentation concise d'objets binaires)
json	JavaScript Notation d'objets (JSON)

Pour de plus amples informations, veuillez consulter API MQTT de mise en service des appareils.

Rubrique	Opérations autorisées du client	Description
$aws/certificates/create/*payload-format*	Publication	Publiez sur cette rubrique pour créer un certificat à partir d'une demande de signature de certificat (CSR), .
$aws/certificates/create/*payload-format* / accepté	S'abonner, recevoir	AWS IoT publie sur cette rubrique après un appel réussi à $aws/certificates/create/*payload-format* .
$aws/certificates/create/*payload-format* / rejeté	S'abonner, recevoir	AWS IoT publie sur cette rubrique après un appel infructueux à $aws/certificates/create/*payload-format* .
$ aws/certificates/create-from-csr/ *payload-format*	Publication	Publie dans cette rubrique pour créer un certificat à partir d'un CSR.
$ aws/certificates/create *payload-format* -from-csr/ /accepté	S'abonner, recevoir	AWS IoT publie dans cette rubrique un appel réussi à $ aws/certificates/create *payload-format* -from-csr/.
$ aws/certificates/create *payload-format* -from-csr/ /rejeté	S'abonner, recevoir	AWS IoT publie dans cette rubrique un appel infructueux à $ aws/certificates/create *payload-format* -from-csr/.
$aws/modèles de provisionnement/ /provision/ *templateName* *payload-format*	Publication	Publiez dans cette rubrique pour enregistrer un objet.
$aws/provisioning-templates/ /provision/ / accepted *templateName* *payload-format*	S'abonner, recevoir	AWS IoT publie sur cette rubrique après un appel réussi à $aws/provisioning-templates/ /provision/*templateName* . *payload-format*
$aws/provisioning-templates/ /provision/ /	S'abonner, recevoir	AWS IoT publie sur cette rubrique après un appel infructueux à $aws/provisioning-

Rubrique	Opérations autorisées du client	Description
rejeté *templateName* *payload-format*		templates/ /provision/*templateName* . *payload-format*

Rubriques de tâche

> **Note**
>
> Les opérations client notées « Recevoir » dans ce tableau indiquent les sujets AWS IoT qui sont publiés directement auprès du client qui les a demandés, que le client soit abonné au sujet ou non. Les clients doivent s'attendre à recevoir ces messages de réponse même s'ils ne s'y sont pas abonnés.
> Ces messages de réponse ne passent pas par le courtier de messages et ne peuvent pas être souscrits par d'autres clients ou règles. Pour vous abonner aux messages relatifs aux activités professionnelles, utilisez les `notify-next` rubriques `notify` et.
> Lorsque vous vous abonnez aux rubriques relatives aux tâches et aux `jobExecution` événements de votre solution de surveillance de flotte, vous devez d'abord activer les événements relatifs aux tâches et à l'exécution des tâches pour recevoir les événements du côté cloud.
> Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.

Rubrique	Opérations autorisées du client	Description
$aws/things/ /jobs/get *thingName*	Publication	Les appareils publient un message dans cette rubrique pour envoyer une demande `GetPendingJobExecutions` . Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/choses//*thingName* jobs/get/accepted	S'abonner, recevoir	Les périphériques s'abonnent à cette rubrique pour recevoir des réponses

Rubrique	Opérations autorisées du client	Description
		positives à une demande `GetPendingJobExecutions` . Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/choses//*thingName* jobs/get/rejected	S'abonner, Recevoir	Les appareils s'abonnent à cette rubrique lorsqu'une demande `GetPendingJobExecutions` est rejetée. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/things/ /jobs/start-next *thingName*	Publication	Les appareils publient un message dans cette rubrique pour envoyer une demande `StartNextPendingJobExecution` . Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/choses//*thingName* jobs/start-next/accepted	S'abonner, recevoir	Les appareils s'abonnent à cette rubrique pour recevoir des réponses positives à une demande `StartNextPendingJobExecution` . Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.

Rubrique	Opérations autorisées du client	Description
$aws/choses//*thingName* jobs/start-next/rejected	S'abonner, Recevoir	Les appareils s'abonnent à cette rubrique lorsqu'une demande `StartNextPendingJobExecution` est rejetée. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/things/ /jobs/ /get *thingName* *jobId*	Publication	Les appareils publient un message dans cette rubrique pour envoyer une demande `DescribeJobExecution`. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/things/ /jobs/ /get/ accepted *thingName* *jobId*	S'abonner, recevoir	Les appareils s'abonnent à cette rubrique pour recevoir des réponses positives à une demande `DescribeJobExecution`. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/things/ /jobs/ /get/rejected *thingName* *jobId*	S'abonner, Recevoir	Les appareils s'abonnent à cette rubrique lorsqu'une demande `DescribeJobExecution` est rejetée. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.

Rubrique	Opérations autorisées du client	Description
$aws/things/ /jobs/ /update *thingName jobId*	Publication	Les appareils publient un message dans cette rubrique pour envoyer une demande `UpdateJobExecution` . Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/things/ /jobs/ /update/ accepted *thingName jobId*	S'abonner, recevoir	Les appareils s'abonnent à cette rubrique pour recevoir des réponses positives à une demande `UpdateJob Execution` . Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches. **Remarque** Seul le périphérique qui publie sur $aws/things/ *thingName* / jobs/ *jobId* /update reçoit des messages à ce sujet.

Rubrique	Opérations autorisées du client	Description
$aws/things/ /jobs/ /update/rejeté *thingName* *jobId*	S'abonner, recevoir	Les appareils s'abonnent à cette rubrique lorsqu'une demande `UpdateJobExecution` est rejetée. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches. **Remarque** Seul le périphérique qui publie sur $aws/things/ *thingName* / jobs/ *jobId* /update reçoit des messages à ce sujet.
$aws/things/ /jobs/notify *thingName*	S'abonner, recevoir	Les appareils s'abonnent à cette rubrique pour recevoir des notifications lorsque l'exécution d'une tâche est ajoutée ou supprimée de la liste des exécutions en attente pour un objet. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.
$aws/things/ /jobs/notify-next *thingName*	S'abonner, recevoir	Les appareils s'abonnent à cette rubrique pour recevoir des notifications lorsque la prochaine tâche en attente d'exécution pour l'objet est modifiée. Pour de plus amples informations, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.

Rubrique	Opérations autorisées du client	Description
$aws/events/job/*jobId*/terminé	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsqu'une tâche est terminée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobId*/annulé	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsqu'une tâche est annulée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobId*/supprimé	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsqu'une tâche est supprimée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobId*/annulation_en cours	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'annulation d'une tâche commence. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/job/*jobId*/deletion_in_en cours	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque la suppression d'une tâche commence. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobId*/réussi	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'exécution d'une tâche est réussie. Pour de plus amples informations, veuillez consulter Événements Jobs.

Rubrique	Opérations autorisées du client	Description
$aws/events/jobExecution/*jobId*/échec	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'exécution d'une tâche échoue. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobId*/rejeté	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'exécution d'une tâche est rejetée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobId*/annulé	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'exécution d'une tâche est annulée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobId*/timed_out	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'exécution d'une tâche arrive à expiration. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobId*/supprimé	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'exécution d'une tâche est retirée. Pour de plus amples informations, veuillez consulter Événements Jobs.
$aws/events/jobExecution/*jobId*/supprimé	S'abonner	Le service Jobs publie un événement sur cette rubrique lorsque l'exécution d'une tâche est supprimée. Pour de plus amples informations, veuillez consulter Événements Jobs.

Rubriques relatives aux commandes

> **Note**
>
> Les opérations client notées « Recevoir » dans ce tableau indiquent les sujets AWS IoT qui sont publiés directement auprès du client qui les a demandés, que le client soit abonné au sujet ou non. Les clients doivent s'attendre à recevoir ces messages de réponse même s'ils ne s'y sont pas abonnés.
> Ces messages de réponse ne passent pas par le courtier de messages et ne peuvent pas être souscrits par d'autres clients ou règles.

Rubrique	Opérations autorisées du client	Description
$aws/commands/// exécutions/ /requête/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/commands/// executions/ /request *<devices> <DeviceID> <ExecutionId>*	S’abonner, recevoir	Les appareils reçoivent un message à ce sujet lorsqu'une demande est faite pour démarrer l'exécution d'une commande depuis la console ou à l'aide de l'`StartCommandExecution` API. Dans ce cas, il *<devices>* peut s'agir d'objets IoT ou de clients MQTT, et il *<DeviceID>* peut s'agir du nom de l'objet IoT ou de l'ID du client MQTT.
$aws/commands///ex écutions/ /response/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>*	Publication	Les appareils utilisent l'API `UpdateCommandExecution` MQTT pour publier un message sur cette rubrique concernant l'exécution des commandes. Le message est publié en réponse à la demande de lancement d'une exécution de commande depuis la console ou à l'aide de l'`StartCommandExecution` API. Le message publié utilisera JSON ou CBOR comme. *<PayloadFormat>*

Rubrique	Opérations autorisées du client	Description
$aws/commands///executions/ /response/accepté/ *<devices> <DeviceID> <ExecutionId> <PayloadFormat>* $aws/commands///executions/ /response/acceptée *<devices> <DeviceID> <ExecutionId>*	S'abonner, recevoir	Si le service cloud a traité avec succès le résultat de l'exécution de la commande, AWS IoT Device Management publie une réponse à la rubrique /accepté.
$aws/commands///executions/ *<devices>* /response/rejeté/ *<DeviceID> <ExecutionId> <PayloadFormat>* $aws/commands///executions/ /response/rejeté *<devices> <DeviceID> <ExecutionId>*	Publication	Si le service cloud n'a pas réussi à traiter le résultat de l'exécution de la commande, AWS IoT Device Management publie une réponse à la rubrique /rejected.

Rubriques de règle

Rubrique	Opérations autorisées du client	Description
$aws/règles/ *ruleName*	Publication	Un appareil ou une application publie dans cette rubrique pour déclencher des règles directement. Pour de plus amples informations, veuillez consulter

Rubrique	Opérations autorisées du client	Description
		Réduction des coûts de messagerie avec Basic Ingest.

Rubriques liées au tunneling sécurisé

Rubrique	Opérations autorisées du client	Description
$aws/things/ /tunnels/notify *thing-name*	S'abonner	AWS IoT publie ce message pour qu'un agent IoT démarre un proxy local sur l'appareil distant. Pour de plus amples informations, veuillez consulter the section called "Extrait de l'agent IoT".

Rubriques de shadow

Les rubriques de cette section sont utilisées par les shadows nommés et non nommés. Les rubriques utilisées par chacun d'eux ne diffèrent que par le préfixe de rubrique. Ce tableau indique le préfixe de rubrique utilisé par chaque type de shadow.

Valeur ***ShadowTopicPrefix***	Type de shadow
$aws/things/ /shadow *thingName*	Shadow non nommé (classique)
$aws/things/ /shadow/nom/ *thingName* *shadowName*	Shadow nommé

Pour créer un sujet complet, sélectionnez le *ShadowTopicPrefix* type d'ombre auquel vous souhaitez faire référence, remplacez-le *thingName* et, le cas échéant*shadowName*, par les valeurs correspondantes, puis ajoutez-le avec le stub du sujet, comme indiqué dans le tableau suivant. N'oubliez pas que les rubriques sont sensibles à la casse.

Rubrique	Opérations autorisées du client	Description
ShadowTopicPrefix / supprimer	Publier/s'abonner	Un appareil ou une application publie dans cette rubrique pour supprimer un shadow. Pour de plus amples informations, veuillez consulter /delete.
ShadowTopicPrefix / supprimer/accepté	S'abonner	Le service Device Shadow envoie des messages à cette rubrique lorsqu'un shadow est supprimé. Pour de plus amples informations, veuillez consulter /delete/accepted.
ShadowTopicPrefix / supprimer/rejeté	S'abonner	Le service Device Shadow envoie des messages à cette rubrique lorsqu'une demande de suppression d'un shadow est rejetée. Pour de plus amples informations, veuillez consulter /delete/rejected.
ShadowTopicPrefix / obtenir	Publier/s'abonner	Une application ou un objet publie un message vide dans cette rubrique pour obtenir un shadow. Pour de plus amples informations, veuillez consulter Rubriques MQTT de Device Shadow.
ShadowTopicPrefix / obtenir/accepter	S'abonner	Le service Device Shadow envoie des messages à cette rubrique lorsqu'une demande de shadow est effectuée avec succès. Pour de plus amples informations, veuillez consulter /get/accepted.
ShadowTopicPrefix / obtenir/rejeté	S'abonner	Le service Device Shadow envoie des messages à cette rubrique lorsqu'une demande de shadow est rejetée. Pour

Rubrique	Opérations autorisées du client	Description
		de plus amples informations, veuillez consulter /get/rejected.
ShadowTopicPrefix / mise à jour	Publier/s'abonner	Un objet ou une application publie dans cette rubrique pour mettre à jour un shadow. Pour de plus amples informati ons, veuillez consulter /update.
ShadowTopicPrefix / mise à jour/accepté	S’abonner	Le service Device Shadow envoie des messages à cette rubrique lorsqu’une mise à jour aété effectuée avec succès dans un shadow. Pour de plus amples informations, veuillez consulter /update/accepted.
ShadowTopicPrefix / mise à jour/rejeté	S’abonner	Le service Device Shadow envoie des messages à cette rubrique lorsqu'un e mise à jour d'un shadow est rejetée. Pour de plus amples informations, veuillez consulter /update/rejected.
ShadowTopicPrefix / mise à jour/delta	S’abonner	Le service Device Shadow envoie des messages à cette rubrique lorsqu'un écart est constaté entre les sections déclarées et les sections souhaitée s d'un shadow. Pour de plus amples informations, veuillez consulter /update/delta.
ShadowTopicPrefix / mise à jour/documents	S’abonner	AWS IoT publie un document d'état sur cette rubrique chaque fois qu'une mise à jour du shadow est effectuée avec succès. Pour de plus amples informations, veuillez consulter /update/documents.

Rubriques de livraison de fichiers basées sur MQTT

> **Note**
>
> Les opérations client notées « Recevoir » dans ce tableau indiquent les sujets AWS IoT qui sont publiés directement auprès du client qui les a demandés, que le client soit abonné au sujet ou non. Les clients doivent s'attendre à recevoir ces messages de réponse même s'ils ne s'y sont pas abonnés. Ces messages de réponse ne passent pas par le courtier de messages et ne peuvent pas être souscrits par d'autres clients ou règles.

Ces messages prennent en charge les tampons de réponse au format CBOR (Concise Binary JavaScript Object Representation) et au format JSON (Object Notation), selon le *payload-format* sujet.

payload-format	Type de données du format de réponse
CBOR	CBOR (Concise Binary Object Representation, représentation concise d'objets binaires)
json	JavaScript Notation d'objets (JSON)

Rubrique	Opérations autorisées du client	Description
$aws/things/ /streams/ /data/ *ThingName StreamId payload-format*	S’abonner, recevoir	AWS La livraison de fichiers basée sur MQTT est publiée dans cette rubrique si la demande « GetStream » d'un appareil est acceptée. La charge utile contient les données du flux. Pour de plus amples informations, veuillez consulter Utilisation de AWS IoT la livraison de fichiers basée sur MQTT sur les appareils.

Rubrique	Opérations autorisées du client	Description
$aws/things/ /streams/ /get/ *ThingName StreamId payload-format*	Publication	Un appareil publie sur cette rubrique pour exécuter une demande GetStream « ». Pour de plus amples informations, veuillez consulter Utilisation de AWS IoT la livraison de fichiers basée sur MQTT sur les appareils.
$aws/things/ /streams/ / description/ *ThingName StreamId payload-format*	S'abonner, recevoir	AWS La livraison de fichiers basée sur MQTT est publiée dans cette rubrique si la demande « DescribeStream » d'un appareil est acceptée. La charge utile contient la description du flux. Pour de plus amples informations, veuillez consulter Utilisation de AWS IoT la livraison de fichiers basée sur MQTT sur les appareils.
$aws/things/ /streams/ / describe/ *ThingName StreamId payload-format*	Publication	Un appareil publie sur cette rubrique pour exécuter une demande DescribeStream « ». Pour de plus amples informations, veuillez consulter Utilisation de AWS IoT la livraison de fichiers basée sur MQTT sur les appareils.

Rubrique	Opérations autorisées du client	Description
$aws/things/ /streams/ / rejeté/ *ThingName StreamId payload-format*	S'abonner, recevoir	AWS La livraison de fichiers basée sur MQTT est publiée dans cette rubrique si une demande « DescribeStream » ou « GetStream » provenant d'un appareil est rejetée. Pour de plus amples informations, veuillez consulter Utilisation de AWS IoT la livraison de fichiers basée sur MQTT sur les appareils.

ARN de sujets réservés

Toutes les rubriques réservées ARNs (Amazon Resource Names) ont la forme suivante :

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

Par exemple, `arn:aws:iot:us-west-2:123EXAMPLE456:topic/$aws/things/thingName/jobs/get/accepted` est un ARN pour le sujet réservé`$aws/things/thingName/jobs/get/accepted`.

Configurations de domaine

Dans AWS IoT Core, vous pouvez utiliser les configurations de domaine pour configurer et gérer les comportements de vos points de terminaison de données. Avec les configurations de domaine, vous pouvez générer plusieurs points de terminaison de AWS IoT Core données, les personnaliser avec vos propres noms de domaine complets (FQDN) et les certificats de serveur associés, et également associer un autorisateur personnalisé. Pour de plus amples informations, veuillez consulter Authentification et autorisation personnalisées.

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans AWS GovCloud (US) Régions AWS.

Dans ce chapitre :

- Qu'est-ce qu'une configuration de domaine ?
- Création et configuration de domaines AWS gérés
- Création et configuration de domaines gérés par le client
- Gestion des configurations de domaine
- Configuration des TLS paramètres dans les configurations de domaine
- Configuration du certificat de serveur pour l'OCSPagrafage

Qu'est-ce qu'une configuration de domaine ?

Dans AWS IoT Core, une configuration de domaine fait référence à l'installation et à la configuration d'un domaine (domaine AWS géré ou domaine géré par le client) pour vos points de terminaison de AWS IoT Core données. AWS IoT Core fournit également un point de terminaison par défaut pour votre AWS compte (`iot:Data-ATS`) avec lequel les appareils peuvent communiquer AWS IoT Core.

Dans cette rubrique :

- Cas d’utilisation
- Concepts clés
- Remarques importantes

Cas d’utilisation

Vous pouvez utiliser les configurations de domaine pour simplifier les tâches suivantes.

- Migrez les appareils vers AWS IoT Core.
- Prenez en charge des parcs d'appareils hétérogènes en maintenant des configurations de domaine distinctes pour des types d'appareils distincts
- Préservez l'identité de marque (par exemple, par le biais d'un nom de domaine) lors de la migration de l'infrastructure d'applications vers AWS IoT Core.

Concepts clés

Les concepts suivants fournissent des informations détaillées sur les configurations de domaine et les concepts associés.

- Configuration du domaine

L'installation et la configuration d'un domaine pour vos AWS IoT Core points de terminaison.

- Domaine de point final par défaut

  Le domaine qui AWS IoT fournit le point de terminaison par défaut tel que`iot:Data-ATS`. Pour trouver le point de terminaison par défaut, exécutez la commande describe-endpoint ou describe-domain-configurationCLI. Vous pouvez également accéder à AWS IoT Core la console et choisir Configurations de domaine dans Connect dans le menu de navigation de gauche. Le point de terminaison par défaut est répertorié avec son nom`iot:Data-ATS`.

- AWS domaine géré

  Le domaine qui AWS gérera. Le choix d'un domaine AWS géré signifie que vos appareils se connecteront à l'aide d'un point de terminaison de données fourni par AWS. AWS gérera le domaine et les certificats.

- Domaine géré par le client

  Le domaine que vous allez gérer. Également connu sous le nom de domaine personnalisé. Le choix d'un domaine géré par le client signifie que vos appareils se connecteront à l'aide d'un point de terminaison de données de domaine personnalisé. Vous allez gérer le domaine et les certificats. Le domaine géré par le client vous permet d'adapter le point URLs de terminaison à vos besoins. Par exemple, vous pouvez utiliser un nom de domaine personnalisé (`your-domain-name.com`) ou appliquer des politiques d'accès spécifiques.

- Type d’authentification

  Type d'authentification que vous choisissez pour authentifier vos appareils lorsque vous vous connectez. AWS IoT Core Lorsque vous créez une configuration de domaine, vous devez spécifier un type d'authentification. Pour de plus amples informations, veuillez consulter ???.

- Protocole d'application

  Les protocoles de couche application que vos appareils utilisent pour se connecter AWS IoT Core. Lorsque vous créez une configuration de domaine, vous devez spécifier un protocole d'application. Pour de plus amples informations, veuillez consulter ???.

Remarques importantes

AWS IoT Core utilise l'TLSextension d'indication du nom du serveur (SNI) pour appliquer les configurations de domaine. Lorsqu'ils connectent des appareils à AWS IoT Core, les clients peuvent envoyer l'extension Server Name Indication (SNI), qui est requise pour les fonctionnalités telles que l'enregistrement multi-comptes, les points de terminaison configurables, les domaines personnalisés et VPC les points de terminaison. Ils doivent également transmettre un nom de serveur identique au nom de domaine que vous spécifiez dans la configuration du domaine. Pour tester ce service, utilisez la version v2 de l'AWS IoT appareil SDKs dans GitHub.

Si vous créez plusieurs points de terminaison de données dans votre Compte AWS, ils partageront AWS IoT Core des ressources telles que des MQTT sujets, des ombres sur les appareils et des règles.

Lorsque vous fournissez les certificats de serveur pour une configuration de domaine AWS IoT Core personnalisée, les certificats ont un maximum de quatre noms de domaine. Pour plus d'informations, consultez Points de terminaison et quotas AWS IoT Core.

Création et configuration de domaines AWS gérés

Vous créez un point de terminaison configurable sur un domaine AWS géré à l'aide du CreateDomainConfigurationAPI. La configuration d'un domaine AWS géré comprend les éléments suivants :

- `domainConfigurationName`

  Un nom défini par l'utilisateur qui identifie la configuration du domaine et dont la valeur doit être unique à votre Région AWS nom. Vous ne pouvez pas utiliser de noms de configuration de domaine commençant par `IoT:`, car ils sont réservés aux points de terminaison par défaut.

- `defaultAuthorizerName` (facultatif)

  Le nom du mécanisme d'autorisation personnalisée à utiliser sur le point de terminaison.

- `allowAuthorizerOverride` (facultatif)

  Valeur booléenne qui indique si les appareils peuvent remplacer l'autorisateur par défaut en spécifiant un autre autorisateur dans l'en-tête de la HTTP demande. Cette valeur est requise si une valeur est spécifiée pour `defaultAuthorizerName`.

- `serviceType` (facultatif)

Type de service fourni par le point de terminaison. AWS IoT Core ne prend en charge que le type de `DATA` service. Lorsque vous spécifiez `DATA`, AWS IoT Core renvoie un point de terminaison avec un type de point de terminaison de `iot:Data-ATS`. Vous ne pouvez pas créer de point de terminaison configurable `iot:Data` (VeriSign).

- `TlsConfig` (facultatif)

  Objet qui spécifie la TLS configuration d'un domaine. Pour de plus amples informations, veuillez consulter ???.

L'exemple de AWS CLI commande suivant crée une configuration de domaine pour un `Data` point de terminaison.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
```

Le résultat de la commande peut ressembler à ce qui suit.

```
{
    "domainConfigurationName": "myDomainConfigurationName",
    "domainConfigurationArn": "arn:aws:iot:us-east-1:123456789012:domainconfiguration/
myDomainConfigurationName/itihw"
}
```

Création et configuration de domaines gérés par le client

Les configurations de domaine vous permettent de spécifier un nom de domaine complet personnalisé (FQDN) auquel vous connecter AWS IoT Core. L'utilisation de domaines gérés par le client (également appelés domaines personnalisés) présente de nombreux avantages : vous pouvez exposer votre propre domaine ou celui de votre entreprise aux clients à des fins de marque ; vous pouvez facilement changer de domaine pour qu'il pointe vers un nouveau courtier ; vous pouvez prendre en charge la mutualisation pour servir les clients ayant différents domaines au sein d'un même domaine Compte AWS ; et vous pouvez gérer les détails de vos propres certificats de serveur, tels que l'autorité de certification racine (CA) utilisée pour signer le certificat, l'algorithme de signature, la profondeur de la chaîne de certificats et le cycle de vie du certificat.

Le flux de travail pour définir une configuration de domaine avec un domaine personnalisé se compose des trois étapes suivantes.

1. Enregistrement de certificats de serveur dans AWS Certificate Manager
2. Création d'une configuration de domaine
3. Création d'DNSenregistrements

Enregistrement des certificats de serveur dans le gestionnaire de AWS certificats

Avant de créer une configuration de domaine avec un domaine personnalisé, vous devez enregistrer votre chaîne de certificats de serveur dans AWS Certificate Manager (ACM). Vous pouvez utiliser les trois types de certificats de serveur suivants.

- Certificats publics générés par ACM
- Certificats externes signés par une autorité de certification publique
- Certificats externes signés par une autorité de certification privée

> **Note**
>
> AWS IoT Core considère qu'un certificat a été signé par une autorité de certification publique s'il est inclus dans le ca-bundle sécurisé de Mozilla.

Exigences du certificat

Consultez la section Conditions préalables à l'importation de certificats pour connaître les exigences relatives à l'importation de certificats dansACM. En plus de ces exigences, AWS IoT Core ajoute les exigences suivantes.

- Le certificat Leaf doit inclure l'extension Extended Key Usage x509 v3 avec la valeur serverAuth(authentification du serveur TLS Web). Si vous demandez le certificat àACM, cette extension est automatiquement ajoutée.
- La profondeur maximale de la chaîne de certificats est de 5 certificats.
- La taille maximale de la chaîne de certificats est de 16 Ko.
- Les algorithmes cryptographiques et les tailles de clé pris en charge incluent RSA 2048 bits (RSA_2048) et ECDSA 256 bits (EC_Prime256v1).

Utiliser un certificat pour plusieurs domaines

Si vous prévoyez d'utiliser un certificat pour couvrir plusieurs sous-domaines, utilisez un domaine générique dans le champ Nom commun (CN) ou Noms alternatifs du sujet (SAN). Par exemple, utilisez **`*.iot.example.com`** pour couvrir dev.iot.example.com, qa.iot.example.com et prod.iot.example.com. Chacune FQDN nécessite sa propre configuration de domaine, mais plusieurs configurations de domaine peuvent utiliser la même valeur générique. Le CN ou le SAN must couvre le domaine FQDN que vous souhaitez utiliser en tant que domaine personnalisé. S'SANsils sont présents, le CN est ignoré et SAN doit couvrir celui FQDN que vous souhaitez utiliser en tant que domaine personnalisé. Cette couverture peut être une correspondance exacte ou une correspondance générique. Une fois qu'un certificat générique a été validé et enregistré sur un compte, les autres comptes de la région ne sont pas autorisés à créer des domaines personnalisés qui se chevauchent avec le certificat.

Les sections suivantes décrivent comment obtenir chaque type de certificat. Chaque ressource de certificat nécessite un Amazon Resource Name (ARN) enregistré sous ACM lequel vous l'utilisez lorsque vous créez la configuration de votre domaine.

ACM-certificats publics générés

Vous pouvez générer un certificat public pour votre domaine personnalisé à l'aide du RequestCertificateAPI. Lorsque vous générez un certificat de cette manière, ACM valide votre propriété du domaine personnalisé. Pour de plus amples informations, veuillez consulter Demander un certificat public dans le AWS Certificate Manager Guide de l'utilisateur.

Certificats externes signés par une autorité de certification publique

Si vous possédez déjà un certificat de serveur signé par une autorité de certification publique (une autorité de certification incluse dans le ca-bundle sécurisé de Mozilla), vous pouvez importer la chaîne de certificats directement dans ACM le. ImportCertificateAPI Pour en savoir plus sur cette tâche et sur les conditions préalables et les exigences en matière de format de certificat, consultez Importation de certificats.

Certificats externes signés par une autorité de certification privée

Si vous disposez déjà d'un certificat de serveur signé par une autorité de certification privée ou auto-signé, vous pouvez utiliser le certificat pour créer la configuration de votre domaine, mais vous devez également créer un certificat public supplémentaire dans ACM pour valider la propriété de votre domaine. Pour ce faire, enregistrez votre chaîne de certificats de serveur à ACM l'aide

du ImportCertificateAPI. Pour en savoir plus sur cette tâche et sur les conditions préalables et les exigences en matière de format de certificat, consultez Importation de certificats.

Création d'un certificat de validation

Après avoir importé votre certificat dansACM, générez un certificat public pour votre domaine personnalisé à l'aide du RequestCertificateAPI. Lorsque vous générez un certificat de cette manière, ACM valide votre propriété du domaine personnalisé. Pour de plus amples informations, veuillez consulter Demander un certificat public. Lorsque vous créez votre configuration de domaine, utilisez ce certificat public comme certificat de validation.

Création d'une configuration de domaine

Vous créez un point de terminaison configurable sur un domaine personnalisé à l'aide du CreateDomainConfigurationAPI. Une configuration de domaine pour un domaine personnalisé se compose des éléments suivants :

- `domainConfigurationName`

  Un nom défini par l'utilisateur qui identifie la configuration du domaine. Les noms de configuration de domaine commençant par `IoT:` sont réservés aux points de terminaison par défaut et ne peuvent pas être utilisés. De plus, cette valeur doit être unique à votre Région AWS.

- `domainName`

  Celui FQDN auquel vos appareils se connectent AWS IoT Core. AWS IoT Core utilise l'TLSextension d'indication du nom du serveur (SNI) pour appliquer les configurations de domaine. Les appareils doivent utiliser cette extension lors de la connexion et transmettre un nom de serveur identique au nom de domaine spécifié dans la configuration de domaine.

- `serverCertificateArns`

  La chaîne ARN de certificats de serveur auprès de laquelle vous vous êtes enregistréACM. AWS IoT Core ne prend actuellement en charge qu'un seul certificat de serveur.

- `validationCertificateArn`

  Le ARN certificat public que vous avez généré ACM pour valider la propriété de votre domaine personnalisé. Cet argument n'est pas requis si vous utilisez un certificat de serveur signé publiquement ou généré par ACM.

- `defaultAuthorizerName (optional)`

Le nom du mécanisme d'autorisation personnalisée à utiliser sur le point de terminaison.

- `allowAuthorizerOverride`

  Valeur booléenne qui indique si les appareils peuvent remplacer l'autorisateur par défaut en spécifiant un autre autorisateur dans l'en-tête de la HTTP demande. Cette valeur est requise si une valeur est spécifiée pour `defaultAuthorizerName`.

- `serviceType`

  AWS IoT Core ne prend actuellement en charge que le type de `DATA` service. Lorsque vous spécifiez`DATA`, AWS IoT renvoie un point de terminaison avec un type de point de terminaison de`iot:Data-ATS`.

- `TlsConfig` (facultatif)

  Objet qui spécifie la TLS configuration d'un domaine. Pour de plus amples informations, veuillez consulter ???.

- `serverCertificateConfig` (facultatif)

  Objet qui spécifie la configuration du certificat de serveur pour un domaine. Pour de plus amples informations, veuillez consulter ???.

La AWS CLI commande suivante crée une configuration de domaine pour iot.example.com.

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" --service-type "DATA"
--domain-name "iot.example.com" --server-certificate-arns serverCertARN --validation-
certificate-arn validationCertArn
```

> **Note**
>
> Une fois que vous avez créé la configuration de votre domaine, il peut s'écouler jusqu'à 60 minutes avant que AWS IoT Core vos certificats de serveur personnalisés ne soient fournis.

Pour de plus amples informations, veuillez consulter ???.

Création d'DNSenregistrements

Après avoir enregistré votre chaîne de certificats de serveur et créé la configuration de votre domaine, créez un DNS enregistrement afin que votre domaine personnalisé pointe vers un AWS IoT domaine. Cet enregistrement doit pointer vers un point de AWS IoT terminaison de type`iot:Data-ATS`. Vous pouvez obtenir votre point de terminaison en utilisant le DescribeEndpointAPI.

La AWS CLI commande suivante montre comment obtenir votre point de terminaison.

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

Une fois que vous avez obtenu votre `iot:Data-ATS` point de terminaison, créez un `CNAME` enregistrement à partir de votre domaine personnalisé pour ce AWS IoT point de terminaison. Si vous créez plusieurs domaines personnalisés dans le même environnement Compte AWS, associez-les à ce même `iot:Data-ATS` point de terminaison.

Résolution des problèmes

Si vous ne parvenez pas à connecter des appareils à un domaine personnalisé, assurez-vous que AWS IoT Core celui-ci a accepté et appliqué votre certificat de serveur. Vous pouvez vérifier que votre certificat AWS IoT Core a été accepté à l'aide de la AWS IoT Core console ou du AWS CLI.

Pour utiliser la AWS IoT Core console, accédez à la page Configurations du domaine et sélectionnez le nom de configuration du domaine. Dans la section Détails du certificat de serveur, vérifiez le statut et les détails du statut. Si le certificat n'est pas valide, remplacez-le ACM par un certificat qui répond aux exigences du certificat répertoriées dans la section précédente. Si le certificat contient le mêmeARN, AWS IoT Core nous le récupérerons et l'appliquerons automatiquement.

Pour vérifier l'état du certificat à l'aide du AWS CLI, appelez le DescribeDomainConfigurationAPIet spécifiez le nom de configuration de votre domaine.

> **Note**
>
> Si votre certificat n'est pas valide, il AWS IoT Core continuera à servir le dernier certificat valide.

Vous pouvez vérifier quel certificat est diffusé sur votre terminal en utilisant la commande openssl suivante.

```
openssl s_client -connect custom-domain-name:8883 -showcerts -servername
custom-domain-name
```

Gestion des configurations de domaine

Cette rubrique couvre les opérations clés qui vous permettent de gérer les ressources de configuration de votre domaine. Vous pouvez également gérer le cycle de vie des configurations existantes à l'aide des méthodes suivantes APIs : ListDomainConfigurations, DescribeDomainConfigurationUpdateDomainConfiguration, et. DeleteDomainConfiguration

Dans cette rubrique :

- Affichage des configurations de domaine
- Mise à jour des configurations de domaine
- Suppression de configurations de domaine
- Rotation des certificats dans des domaines personnalisés

Affichage des configurations de domaine

Pour renvoyer une liste paginée de toutes les configurations de domaine de votre Compte AWS, utilisez le ListDomainConfigurationsAPI. Vous pouvez consulter les détails d'une configuration de domaine particulière à l'aide du DescribeDomainConfigurationAPI. Cela API prend un seul `domainConfigurationName` paramètre et renvoie les détails de la configuration spécifiée.

Exemple

Mise à jour des configurations de domaine

Pour mettre à jour le statut ou l'autorisateur personnalisé de la configuration de votre domaine, utilisez le UpdateDomainConfigurationAPI. Vous pouvez définir l'état sur `ENABLED` ou sur `DISABLED`. Si vous désactivez la configuration du domaine, les appareils connectés à ce domaine reçoivent une erreur d'authentification. Actuellement, vous ne pouvez pas mettre à jour le certificat de serveur dans la configuration de votre domaine. Pour modifier le certificat d'une configuration de domaine, vous devez le supprimer, puis le recréer.

Exemple

Suppression de configurations de domaine

Avant de supprimer une configuration de domaine, utilisez le UpdateDomainConfigurationAPIpour définir le statut sur`DISABLED`. Vous évitez ainsi de supprimer accidentellement le point de

terminaison. Après avoir désactivé la configuration du domaine, supprimez-la à l'aide du DeleteDomainConfigurationAPI. Vous devez placer les domaines AWS gérés en `DISABLED` statut pendant 7 jours avant de pouvoir les supprimer. Vous pouvez placer des domaines personnalisés dans un `DISABLED` statut, puis les supprimer immédiatement.

Exemple

Une fois que vous avez supprimé une configuration de domaine, le certificat de serveur associé à ce domaine personnalisé AWS IoT Core n'est plus diffusé.

Rotation des certificats dans des domaines personnalisés

Vous devrez peut-être remplacer régulièrement votre certificat de serveur par un certificat mis à jour. Le tarif auquel vous le faites dépend de la durée de validité de votre certificat. Si vous avez généré votre certificat de serveur à l'aide de AWS Certificate Manager (ACM), vous pouvez configurer le certificat pour qu'il soit renouvelé automatiquement. Lorsque vous ACM renouvelez votre certificat, il récupère AWS IoT Core automatiquement le nouveau certificat. Vous n'avez aucune action supplémentaire à effectuer. Si vous avez importé votre certificat de serveur depuis une autre source, vous pouvez le faire pivoter en le réimportant dans. ACM Pour plus d'informations sur la réimportation de certificats, voir Réimporter un certificat.

> **Note**
>
> AWS IoT Core ne récupère les mises à jour des certificats que dans les conditions suivantes.
>
> - Le nouveau certificat est ARN identique à l'ancien.
> - Le nouveau certificat a le même algorithme de signature, le même nom commun ou le même nom alternatif de sujet que l'ancien.

Configuration des TLS paramètres dans les configurations de domaine

AWS IoT Core fournit des politiques de sécurité prédéfinies vous permettant de personnaliser vos paramètres Transport Layer Security (TLS) pour les versions TLS1.2 et TLS1.3 dans les configurations de domaine. Une politique de sécurité est une combinaison de TLS protocoles et de leurs chiffrements qui déterminent les protocoles et chiffrements pris en charge lors des TLS négociations entre un client et un serveur. Grâce aux politiques de sécurité prises en charge, vous pouvez gérer les TLS paramètres de vos appareils avec plus de flexibilité, appliquer le plus grand

nombre de mesures de up-to-date sécurité lors de la connexion de nouveaux appareils et maintenir des TLS configurations cohérentes pour les appareils existants.

Le tableau suivant décrit les politiques de sécurité, leurs TLS versions et les régions prises en charge :

Nom de la politique de sécurité	Soutenu Régions AWS
oTSecurityPolitique en matière_ _1_3_2022_10 TLS13	Tout Régions AWS
oTSecurityPolitique en matière_ _1_2_2022_10 TLS13	Tout Régions AWS
oTSecurityPolitique en matière_ _1_2_2022_10 TLS12	Tout Régions AWS
oTSecurityPolitique en matière_ _1_0_2016_01 TLS12	ap-east-1, ap-northeast-2, ap-sud-1, ap-southeast-2, ca-central-1, cn-nord-1, cn-northwest-1, cn-northwest-1, eu-nord-1, eu-west-2, eu-west-3, me-south-1, sa-east-1, us-east-2, us-west-1
oTSecurityPolitique en matière_ _1_0_2015_01 TLS12	ap-northeast-1, ap-southeast-1, eu-central-1, eu-west-1, us-east-1, us-west-2

Les noms des politiques de sécurité AWS IoT Core incluent des informations de version basées sur l'année et le mois de leur publication. Si vous créez un nouveau domaine, il n'est pas nécessaire de mettre à niveau `IoTSecurityPolicy_TLS13_1_2_2022_10`. Pour un tableau complet des politiques de sécurité avec des détails sur les protocoles, TCP les ports et les chiffrements, consultez la section Politiques de sécurité. AWS IoT Core ne prend pas en charge les politiques de sécurité personnalisées. Pour de plus amples informations, veuillez consulter ???.

Pour configurer TLS les paramètres des configurations de domaine, vous pouvez utiliser la AWS IoT console ou le AWS CLI.

Table des matières

- Configurer TLS les paramètres dans les configurations de domaine (console)
- Configurer TLS les paramètres dans les configurations de domaine (CLI)

Configurer TLS les paramètres dans les configurations de domaine (console)

Pour configurer les TLS paramètres à l'aide de la AWS IoT console

1. Connectez-vous à la AWS IoT console AWS Management Console et ouvrez-la.
2. Pour configurer TLS les paramètres lorsque vous créez une nouvelle configuration de domaine, procédez comme suit.
    1. Dans le volet de navigation de gauche, choisissez Paramètres, puis, dans la section Configurations de domaine, choisissez Créer une configuration de domaine.
    2. Sur la page Créer une configuration de domaine, dans la section Paramètres de domaine personnalisés - facultatif, choisissez une politique de sécurité dans Sélectionner une stratégie de sécurité.
    3. Suivez le widget et terminez le reste des étapes. Choisissez Créer une configuration de domaine.
3. Pour mettre à jour TLS les paramètres d'une configuration de domaine existante, procédez comme suit.
    1. Dans le volet de navigation de gauche, choisissez Paramètres, puis, sous Configurations de domaine, choisissez une configuration de domaine.
    2. Sur la page des détails de configuration du domaine, choisissez Modifier. Ensuite, dans la section Paramètres de domaine personnalisés - facultatif, sous Sélectionner une politique de sécurité, choisissez une politique de sécurité.
    3. Choisissez Mettre à jour la configuration du domaine.

Pour plus d'informations, consultez les sections Création d'une configuration de domaine et Gestion des configurations de domaine.

Configurer TLS les paramètres dans les configurations de domaine (CLI)

Vous pouvez utiliser les update-domain-configurationCLIcommandes create-domain-configurationet pour configurer vos TLS paramètres dans les configurations de domaine.

1. Pour définir TLS les paramètres à l'aide de la create-domain-configurationCLIcommande :

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

   Le résultat de cette commande peut ressembler à ce qui suit :

```
{
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

   Si vous créez une nouvelle configuration de domaine sans spécifier la politique de sécurité, la valeur par défaut sera : `IoTSecurityPolicy_TLS13_1_2_2022_10`.

2. Pour décrire les TLS paramètres à l'aide de la describe-domain-configurationCLIcommande :

```
aws iot describe-domain-configuration \
    --domain-configuration-name domainConfigurationName
```

   Cette commande peut renvoyer les détails de configuration du domaine qui incluent les TLS paramètres suivants :

```
{
 "tlsConfig": {
 "securityPolicy": "IoTSecurityPolicy_TLS13_1_2_2022_10"
 },
 "domainConfigurationStatus": "ENABLED",
 "serviceType": "DATA",
 "domainType": "AWS_MANAGED",
 "domainName": "d1234567890abcdefghij-ats.iot.us-west-2.amazonaws.com",
 "serverCertificates": [],
 "lastStatusChangeDate": 1678750928.997,
 "domainConfigurationName": "test",
 "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

3. Pour mettre à jour TLS les paramètres à l'aide de la update-domain-configurationCLIcommande :

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

Le résultat de cette commande peut ressembler à ce qui suit :

```
{
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

4. Pour mettre à jour les TLS paramètres de votre ATS terminal, exécutez la update-domain-configurationCLIcommande. Le nom de configuration de domaine de votre ATS point de terminaison est`iot:Data-ATS`.

```
aws iot update-domain-configuration \
    --domain-configuration-name "iot:Data-ATS" \
    --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

La sortie de la commande ressemble à ce qui suit :

```
{
"domainConfigurationName": "iot:Data-ATS",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
iot:Data-ATS"
}
```

Pour plus d'informations, voir CreateDomainConfigurationet UpdateDomainConfigurationdans la AWS APIréférence.

Configuration du certificat de serveur pour l'OCSPagrafage

AWS IoT Core prend en charge l'agrafage du protocole Online Certificate Status Protocol (OCSP) pour les certificats de serveur, également appelé agrafage ou OCSP agrafage de certificats de serveur. OCSP Il s'agit d'un mécanisme de sécurité utilisé pour vérifier l'état de révocation du certificat de serveur lors d'une poignée de main Transport Layer Security (TLS). OCSPl'agrafage

vous AWS IoT Core permet d'ajouter une couche de vérification supplémentaire à la validité du certificat de serveur de votre domaine personnalisé.

Vous pouvez activer l'OCSPagrafage des certificats de serveur AWS IoT Core pour vérifier la validité du certificat en interrogeant régulièrement le OCSP répondeur. Le paramètre OCSP d'agrafage fait partie du processus de création ou de mise à jour d'une configuration de domaine avec un domaine personnalisé. OCSPl'agrafage vérifie en permanence l'état de révocation du certificat du serveur. Cela permet de vérifier que les certificats révoqués par l'autorité de certification ne sont plus approuvés par les clients qui se connectent à vos domaines personnalisés. Pour de plus amples informations, veuillez consulter ???.

L'OCSPagrafage des certificats de serveur permet de vérifier l'état de révocation en temps réel, de réduire la latence associée à la vérification de l'état de révocation et d'améliorer la confidentialité et la fiabilité des connexions sécurisées. Pour plus d'informations sur les avantages de l'OCSPagrafage, consultez. ???

> **Note**
>
> Cette fonctionnalité n'est pas disponible dans AWS GovCloud (US) Regions.

Dans cette rubrique :

- Qu’est-ce qu’OCSP ?
- Comment fonctionne OCSP l'agrafage
- Activation du certificat de serveur OCSP dans AWS IoT Core
- Configuration du certificat de serveur OCSP pour les points de terminaison privés dans AWS IoT Core
- Remarques importantes concernant l'utilisation de l'OCSPagrafage de certificats de serveur dans AWS IoT Core
- Résolution des problèmes liés à l'OCSPagrafage des certificats de serveur AWS IoT Core

Qu’est-ce qu’OCSP ?

Le protocole d'état des certificats en ligne (OCSP) permet de fournir le statut de révocation d'un certificat de serveur pour une poignée de main avec Transport Layer Security (TLS).

Concepts clés

Les concepts clés suivants fournissent des détails sur le protocole d'état des certificats en ligne (OCSP).

OCSP

OCSPest utilisé pour vérifier l'état de révocation du certificat lors de la poignée de main Transport Layer Security (TLS). OCSPpermet la validation en temps réel des certificats. Cela confirme que le certificat n'a pas été révoqué ou n'a pas expiré depuis son émission. OCSPest également plus évolutif que les listes de révocation de certificats traditionnelles (CRLs). OCSPles réponses sont plus petites et peuvent être générées efficacement, ce qui les rend plus adaptées aux infrastructures à clé privée à grande échelle (PKIs).

OCSPrépondeur

Un OCSP répondeur (également appelé OCSP serveur) reçoit et répond aux OCSP demandes des clients qui cherchent à vérifier l'état de révocation des certificats.

Côté client OCSP

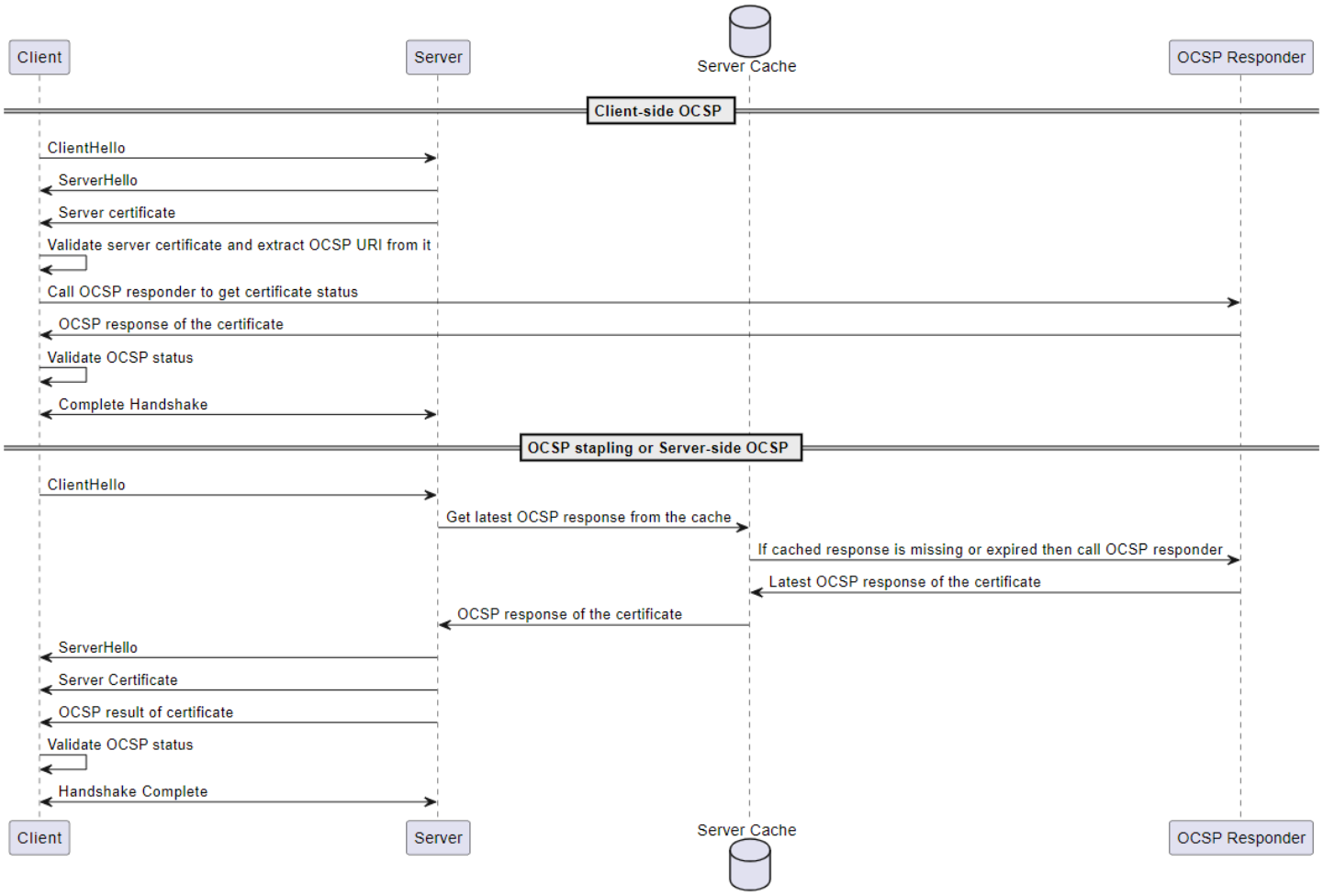
Côté clientOCSP, le client a l'habitude de contacter un OCSP répondeur OCSP pour vérifier l'état de révocation du certificat lors de la poignée de main. TLS

Côté serveur OCSP

Côté serveur OCSP (également connu sous le nom d'OCSPagrafage), le serveur est activé (plutôt que le client) pour envoyer la demande au répondeur. OCSP Le serveur agrafe la OCSP réponse au certificat et la renvoie au client lors de la TLS poignée de main.

OCSPdiagrammes

Le schéma suivant illustre le fonctionnement côté client OCSP et côté serveurOCSP.



Côté client OCSP

1. Le client envoie un `ClientHello` message pour initier la prise TLS de contact avec le serveur.
2. Le serveur reçoit le message et y répond par un `ServerHello` message. Le serveur envoie également le certificat du serveur au client.
3. Le client valide le certificat du serveur et en OCSP URI extrait un.
4. Le client envoie une demande de vérification de révocation de certificat au OCSP répondeur.
5. Le OCSP répondeur envoie une OCSP réponse.
6. Le client valide le statut du certificat à partir de la OCSP réponse.
7. La TLS poignée de main est terminée.

Côté serveur OCSP

1. Le client envoie un `ClientHello` message pour initier la prise TLS de contact avec le serveur.

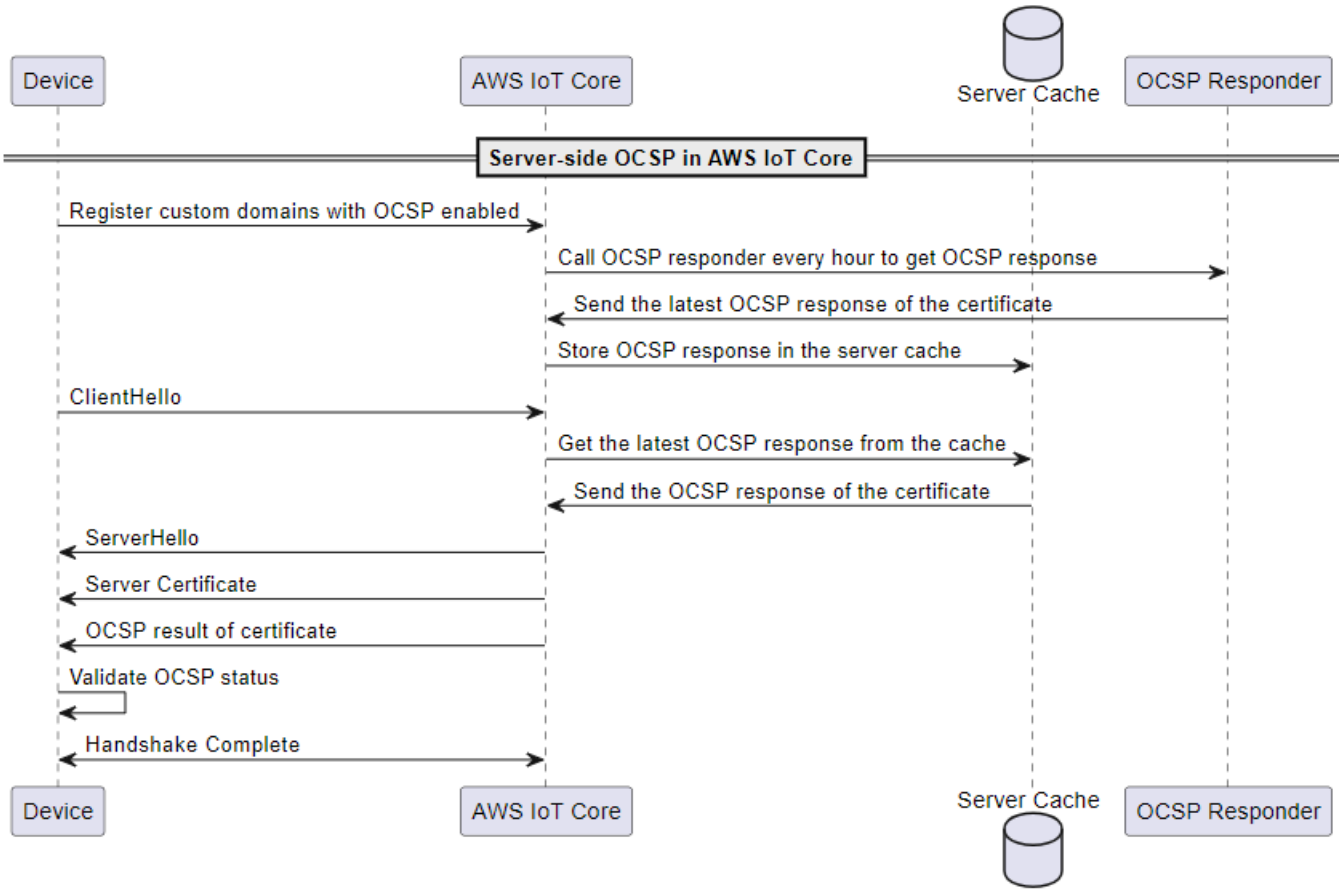
2. Le serveur reçoit le message et obtient la dernière OCSP réponse mise en cache. Si la réponse mise en cache est manquante ou a expiré, le serveur appellera le OCSP répondeur pour connaître l'état du certificat.
3. Le OCSP répondeur envoie une OCSP réponse au serveur.
4. Le serveur envoie un `ServerHello` message. Le serveur envoie également le certificat du serveur et le statut du certificat au client.
5. Le client valide le statut du OCSP certificat.
6. La TLS poignée de main est terminée.

Comment fonctionne OCSP l'agrafage

OCSPl'agrafage est utilisé lors de la prise de TLS contact entre le client et le serveur pour vérifier l'état de révocation du certificat du serveur. Le serveur fait la OCSP demande au OCSP répondeur et agrafe les OCSP réponses aux certificats renvoyés au client. En demandant au serveur de faire la demande au OCSP répondeur, les réponses peuvent être mises en cache puis utilisées plusieurs fois pour de nombreux clients.

Comment fonctionne OCSP l'agrafage dans AWS IoT Core

Le schéma suivant montre le fonctionnement de l'OCSPagrafage côté serveur. AWS IoT Core



1. L'appareil doit être enregistré avec des domaines personnalisés avec l'OCSPagrafage activé.
2. AWS IoT Core appelle le OCSP répondeur toutes les heures pour connaître l'état du certificat.
3. Le OCSP répondeur reçoit la demande, envoie la dernière OCSP réponse et stocke la réponse mise en cacheOCSP.
4. L'appareil envoie un `ClientHello` message pour initier la TLS poignée de main AWS IoT Core.
5. AWS IoT Core obtient la dernière OCSP réponse du cache du serveur, qui répond par une OCSP réponse du certificat.
6. Le serveur envoie un `ServerHello` message à l'appareil. Le serveur envoie également le certificat du serveur et le statut du certificat au client.
7. L'appareil valide le statut du OCSP certificat.
8. La TLS poignée de main est terminée.

Avantages de l'OCSPagrafage par rapport aux contrôles côté client OCSP

Parmi les avantages de l'OCSPagrafage de certificats de serveur, citons les suivants :

Confidentialité améliorée

Sans OCSP agrafage, l'appareil du client peut exposer des informations à des OCSP intervenants tiers, ce qui peut compromettre la confidentialité des utilisateurs. OCSPl'agrafage atténue ce problème en permettant au serveur d'obtenir la OCSP réponse et de la transmettre directement au client.

Fiabilité améliorée

OCSPl'agrafage peut améliorer la fiabilité des connexions sécurisées car il réduit le risque de panne des OCSP serveurs. Lorsque OCSP les réponses sont agrafées, le serveur inclut la réponse la plus récente avec le certificat. Cela permet aux clients d'accéder au statut de révocation même si le OCSP répondeur est temporairement indisponible. OCSPl'agrafage permet d'atténuer ces problèmes car le serveur récupère les OCSP réponses périodiquement et inclut les réponses mises en cache dans la poignée de main. TLS Cela réduit la dépendance à l'égard de la disponibilité en temps réel des OCSP intervenants.

Charge de serveur réduite

OCSPl'agrafage décharge le serveur de la charge de répondre aux OCSP demandes des OCSP répondeurs. Cela permet de répartir la charge de manière plus uniforme, ce qui rend le processus de validation des certificats plus efficace et évolutif.

Latence réduite

OCSPl'agrafage réduit le temps de latence associé à la vérification de l'état de révocation d'un certificat lors de la poignée de main. TLS Au lieu que le client doive interroger un OCSP serveur séparément, le serveur envoie la demande et joint la OCSP réponse au certificat du serveur lors de la prise de contact.

Activation du certificat de serveur OCSP dans AWS IoT Core

Pour activer l'OCSPagrafage des certificats de serveur AWS IoT Core, créez une configuration de domaine pour un domaine personnalisé ou mettez à jour une configuration de domaine personnalisée existante. Pour obtenir des informations générales sur la création d'une configuration de domaine avec un domaine personnalisé, consultez???.

Suivez les instructions ci-dessous pour activer l'agrafage OCSP du serveur à l'aide AWS Management Console de ou. AWS CLI

Console

Pour activer l'OCSPagrafage des certificats de serveur à l'aide de la AWS IoT console :

1. Dans le menu de navigation, choisissez Paramètres, puis choisissez Créer une configuration de domaine, ou choisissez une configuration de domaine existante pour un domaine personnalisé.
2. Si vous avez choisi de créer une nouvelle configuration de domaine à l'étape précédente, vous verrez la page Créer une configuration de domaine. Dans la section Propriétés de configuration du domaine, sélectionnez Domaine personnalisé. Entrez les informations pour créer une configuration de domaine.

   Si vous choisissez de mettre à jour une configuration de domaine existante pour un domaine personnalisé, vous verrez la page des détails de configuration du domaine. Choisissez Modifier.
3. Pour activer l'agrafage OCSP du serveur, choisissez Activer l'agrafage du certificat du serveur dans la OCSP sous-section Configurations des certificats du serveur.
4. Choisissez Créer une configuration de domaine ou Mettre à jour la configuration de domaine.

AWS CLI

Pour activer l'OCSPagrafage de certificats de serveur à l'aide de : AWS CLI

1. Si vous créez une nouvelle configuration de domaine pour un domaine personnalisé, la commande permettant d'activer l'agrafage OCSP du serveur peut se présenter comme suit :

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true|false"
```

2. Si vous mettez à jour une configuration de domaine existante pour un domaine personnalisé, la commande permettant d'activer l'agrafage OCSP du serveur peut se présenter comme suit :

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
```

```
        --server-certificate-config "enableOCSPCheck=true|false"
```

Pour plus d'informations, voir CreateDomainConfigurationet UpdateDomainConfigurationà partir de la AWS IoT API référence.

Configuration du certificat de serveur OCSP pour les points de terminaison privés dans AWS IoT Core

OCSPpour les points de terminaison privés vous permet d'utiliser vos OCSP ressources privées au sein de votre Amazon Virtual Private Cloud (AmazonVPC) pour vos AWS IoT Core opérations. Le processus implique la mise en place d'une fonction Lambda qui agit comme un OCSP répondeur. La fonction Lambda peut utiliser vos OCSP ressources privées pour élaborer des OCSP réponses qui AWS IoT Core utiliseront.

Fonction Lambda

Avant de configurer le serveur OCSP pour un point de terminaison privé, créez une fonction Lambda qui agit comme un répondeur conforme au protocole d'état du certificat en ligne (RFC) Request for Comments () 6960, prenant en charge OCSP les réponses de base. OCSP La fonction Lambda accepte un codage base64 de la OCSP demande au format Distinguished Encoding Rules (). DER La réponse de la fonction Lambda est également une réponse codée en base64 OCSP au format. DER La taille de la réponse ne doit pas dépasser 4 kilo-octets (KiB). La fonction Lambda doit être identique à la Compte AWS configuration Région AWS du domaine. Voici des exemples de fonctions Lambda.

Exemples de fonctions Lambda

JavaScript

```
import * as pkijs from 'pkijs';
console.log('Loading function');

export const handler = async (event, context) => {
    const requestBytes = decodeBase64(event);
    const ocspRequest = pkijs.OCSPRequest.fromBER(requestBytes);

    console.log("Here is a better look at the OCSP request");
    console.log(ocspRequest.toJSON());

    const ocspResponse = getOcspResponse();
```

```
    console.log("Here is a better look at the OCSP response");
    console.log(ocspResponse.toJSON());

   const responseBytes = ocspResponse.toSchema().toBER();
   return encodeBase64(responseBytes);
};

function getOcspResponse() {
    const responseString = "MIIC/
woBAKCCAvgwggL0BgkrBgEFBQcwAQEEggLlMIIC4TCByqFkMGIxJzAlBgNVBAoMHlJpY2hhcmQncyBEaXNjb3VudCBMY
p5w7WOtPjp3otNtVgIBAYAAGA8yMDI0MDQyMzE4NTMyNVowDQYJKoZIhvcNAQELBQADggIBAJFRyjDAHfazNejo704Ra
+s82R1spDarr3k7Pzkod9jJhwsZ2YgushlS4Npfe4lHCdwFyZR75WXrW55aXFddy03KLz01ZLNYyxkleW3f5dgrUcRU3
DEBiyS7ZsyhKo6igWU/SY7YMSKgwBvFsqSDcOa/hRYQkxWKWJ19gcz8CIkWN7NvfIxCs6VrAdzEJwmE7y3v
+jdfhxW9JmI4xStE4K0tAR9vVOOfKs7NvxXj7oc9pCSG60xl96kaEE6PaY1YsfNTsKQ7pyCJ0s7/2q
+ieZ4AtNyzw1XBadPzPJNv6E0LvI24yQZqN5wACvtut5prMMRxAHbOy
+abLZR58wloFSELtGJ7UD96LFv1GgtC5s
+2QlzPc4bEEof7Lo1EISt3j2ibNch8LxhqTQ4ufrbhsMkpSOTFYEJVMJF6aKj/OGXBUUqgc0Jx6jjJXNQd
+l5KCY9pQFeb/wVUYC6mYqZOkNNMMJxPbHHbFnqb68yO+g5BE9011N44YXoPVJYoXxBLFX+OpRu9cqPkT9/
vlkKd+SYXQknwZ81agKzhf1HsBKabtJwNVMlBKaI8g5UGa7Bxi6ewH3ezdWiERRUK7F56OM53wto/";
    const responseBytes = decodeBase64(responseString);
    return pkijs.OCSPResponse.fromBER(responseBytes);
}

function decodeBase64(input) {
    const binaryString = atob(input);

    const byteArray = new Uint8Array(binaryString.length);
    for (var i = 0; i < binaryString.length; i++) {
        byteArray[i] = binaryString.charCodeAt(i);
    }

    return byteArray.buffer;
}

function encodeBase64(buffer) {
    var binary = '';
    const bytes = new Uint8Array( buffer );
    const len = bytes.byteLength;

    for (var i = 0; i < len; i++) {
        binary += String.fromCharCode( bytes[ i ] );
    }
```

```
    return btoa(binary);
}
```

Java

```
package com.example.ocsp.responder;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import org.bouncycastle.cert.ocsp.OCSPReq;
import org.bouncycastle.cert.ocsp.OCSPResp;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Base64;

public class LambdaResponderApplication implements RequestHandler<String, String> {
    @Override
    public String handleRequest(final String input, final Context context) {
        LambdaLogger logger = context.getLogger();

        byte[] decodedInput = Base64.getDecoder().decode(input);

        OCSPReq req;
        try {
            req = new OCSPReq(decodedInput);
        } catch (IOException e) {
            logger.log("Got an IOException creating the OCSP request: " +
 e.getMessage());
            throw new RuntimeException(e);
        }

        try {
            OCSPResp response = businessLogic.getMyResponse();
            String toReturn =
 Base64.getEncoder().encodeToString(response.getEncoded());
            return toReturn;
        } catch (Exception e) {
            logger.log("Got an exception creating the response: " + e.getMessage());
            return "";
        }
    }
}
```

Autorisation d' AWS IoT invoquer votre fonction Lambda

Lors de la création de la configuration du domaine avec un OCSP répondeur Lambda, vous devez AWS IoT autoriser l'appel de la fonction Lambda une fois celle-ci créée. Pour accorder l'autorisation, vous pouvez utiliser la CLI commande add permission.

Accordez l'autorisation à votre fonction Lambda à l'aide du AWS CLI

1. Après avoir inséré vos valeurs, entrez la commande suivante. Notez que la valeur `statement-id` doit être unique. Remplacez *`Id-1234`* par la valeur exacte que vous avez, sinon vous risquez de recevoir une `ResourceConflictException` erreur.

   ```
   aws lambda add-permission  \
   --function-name "ocsp-function" \
   --principal "iot.amazonaws.com" \
   --action "lambda:InvokeFunction" \
   --statement-id "Id-1234" \
   --source-arn arn:aws:iot:us-east-1:123456789012:domainconfiguration/<domain-config-name>/*
   --source-account 123456789012
   ```

   La configuration du domaine IoT ARNs suivra le schéma suivant. Le suffixe généré par le service ne sera pas connu avant la création, vous devez donc le remplacer par un. * Vous pouvez mettre à jour l'autorisation une fois que la configuration du domaine a été créée et que l'exactitude ARN est connue.

   `arn:aws:iot:use-east-1:123456789012:domainconfiguration/domain-config-name/service-generated-suffix`

2. Si la commande aboutit, elle renvoie une instruction d'autorisation, comme dans cet exemple. Vous pouvez passer à la section suivante pour configurer l'OCSPagrafage pour les points de terminaison privés.

   ```
   {
       "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
   \":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
   \",\"Resource\":\"arn:aws:lambda:us-east-1:123456789012:function:ocsp-
   function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:iot:us-
   east-1:123456789012:domainconfiguration/domain-config-name/*\"}}}"
   }
   ```

Si la commande échoue, elle renvoie une erreur, comme dans cet exemple. Vous devez vérifier et corriger l'erreur avant de continuer.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:us-east-1:123456789012:function:ocsp-function
```

Configuration de l'OCSPagrafage du serveur pour les points de terminaison privés

Console

Pour configurer l'OCSPagrafage des certificats de serveur à l'aide de la AWS IoT console :

1. Dans le menu de navigation, choisissez Paramètres, puis choisissez Créer une configuration de domaine, ou choisissez une configuration de domaine existante pour un domaine personnalisé.
2. Si vous avez choisi de créer une nouvelle configuration de domaine à l'étape précédente, vous verrez la page Créer une configuration de domaine. Dans la section Propriétés de configuration du domaine, sélectionnez Domaine personnalisé. Entrez les informations pour créer une configuration de domaine.

   Si vous choisissez de mettre à jour une configuration de domaine existante pour un domaine personnalisé, vous verrez la page des détails de configuration du domaine. Choisissez Modifier.
3. Pour activer l'agrafage OCSP du serveur, choisissez Activer l'agrafage du certificat du serveur dans la OCSP sous-section Configurations des certificats du serveur.
4. Choisissez Créer une configuration de domaine ou Mettre à jour la configuration de domaine.

AWS CLI

Pour configurer l'OCSPagrafage des certificats de serveur à l'aide de : AWS CLI

1. Si vous créez une nouvelle configuration de domaine pour un domaine personnalisé, la commande permettant de configurer le certificat de serveur OCSP pour les points de terminaison privés peut se présenter comme suit :

```
aws iot create-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
```

```
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

2. Si vous mettez à jour une configuration de domaine existante pour un domaine personnalisé, la commande permettant de configurer le certificat de serveur OCSP pour les points de terminaison privés peut se présenter comme suit :

```
aws iot update-domain-configuration --domain-configuration-name
 "myDomainConfigurationName" \
        --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
        --server-certificate-config "enableOCSPCheck=true,
 ocspAuthorizedResponderArn=arn:aws:acm:us-
east-1:123456789012:certificate/certificate_ID, ocspLambdaArn=arn:aws:lambda:us-
east-1:123456789012:function:my-function"
```

enableOCSPCheck

Il s'agit d'une valeur booléenne qui indique si le contrôle d'OCSPagrafage du serveur est activé ou non. Pour activer l'OCSPagrafage des certificats de serveur, cette valeur doit être vraie.

ocspAuthorizedResponderArn

Il s'agit d'une valeur de chaîne du nom de ressource Amazon (ARN) pour un certificat X.509 stocké dans AWS Certificate Manager (ACM). S'il est fourni, AWS IoT Core il utilisera ce certificat pour valider la signature de la OCSP réponse reçue. S'il n'est pas fourni, AWS IoT Core il utilisera le certificat émetteur pour valider les réponses. Le certificat doit être identique Compte AWS à la configuration du domaine. Région AWS Pour plus d'informations sur l'enregistrement de votre certificat de répondeur autorisé, voir Importer des certificats dans AWS Certificate Manager.

ocspLambdaArn

Il s'agit d'une valeur de chaîne de l'Amazon Resource Name (ARN) pour une fonction Lambda qui agit comme un répondeur conforme à la norme Request for Comments (RFC) 6960 () et prend en charge OCSP les réponses de base. OCSP La fonction Lambda accepte un codage base64

de la OCSP demande qui est codé selon le format. DER La réponse de la fonction Lambda est également une réponse codée en base64 OCSP au format. DER La taille de la réponse ne doit pas dépasser 4 kilo-octets (KiB). La fonction Lambda doit être identique à la Compte AWS configuration Région AWS du domaine.

Pour plus d'informations, voir CreateDomainConfigurationet UpdateDomainConfigurationà partir de la AWS IoT API référence.

Remarques importantes concernant l'utilisation de l'OCSPagrafage de certificats de serveur dans AWS IoT Core

Lorsque vous utilisez un certificat de serveur OCSP dans AWS IoT Core, gardez à l'esprit les points suivants :

1. AWS IoT Core ne prend en charge que OCSP les répondeurs accessibles via des IPv4 adresses publiques.
2. La fonction d'OCSPagrafage intégrée AWS IoT Core ne prend pas en charge le répondeur autorisé. Toutes les OCSP réponses doivent être signées par l'autorité de certification qui a signé le certificat, et l'autorité de certification doit faire partie de la chaîne de certificats du domaine personnalisé.
3. La fonction d'OCSPagrafage intégrée AWS IoT Core ne prend pas en charge les domaines personnalisés créés à l'aide de certificats auto-signés.
4. AWS IoT Core appelle un OCSP répondeur toutes les heures et met la réponse en cache. Si l'appel au répondeur échoue, AWS IoT Core agrafera la réponse valide la plus récente.
5. Si elle n'`nextUpdateTime`est plus valide, la réponse AWS IoT Core sera supprimée du cache et la TLS poignée de mains n'inclura pas les données de OCSP réponse avant le prochain appel réussi au OCSP répondeur. Cela peut se produire lorsque la réponse mise en cache a expiré avant que le serveur ne reçoive une réponse valide du OCSP répondeur. La valeur de `nextUpdateTime` suggère que la OCSP réponse sera valide jusqu'à ce moment. Pour plus d'informations sur `nextUpdateTime`, consultez ???.
6. Parfois, AWS IoT Core ne reçoit pas la OCSP réponse ou supprime la OCSP réponse existante parce qu'elle a expiré. Si de telles situations se produisent, AWS IoT Core continuera à utiliser le certificat de serveur fourni par le domaine personnalisé sans OCSP réponse.
7. La taille de la OCSP réponse ne peut pas dépasser 4 KiB.

Résolution des problèmes liés à l'OCSPagrafage des certificats de serveur AWS IoT Core

AWS IoT Core envoie la `RetrieveOCSPStapleData.Success` métrique et les entrées du `RetrieveOCSPStapleData` journal à CloudWatch. La métrique et les entrées du journal peuvent aider à détecter les problèmes liés à la récupération OCSP des réponses. Pour plus d'informations, consultez ??? et ???.

Connectez-vous aux points de AWS IoT terminaison FIPS

AWS IoT fournit des points de terminaison compatibles avec la norme fédérale de traitement de l'information (FIPS) 140-2. Les points de terminaison conformes à la norme FIPS sont différents des points de terminaison standard. AWS Pour interagir avec AWS IoT de manière conforme à FIPS, vous devez utiliser les points de terminaison décrits ci-dessous avec votre client conforme à FIPS. La AWS IoT console n'est pas conforme à la norme FIPS.

Les sections suivantes décrivent comment accéder aux AWS IoT points de terminaison conformes à la norme FIPS à l'aide de l'API REST, d'un SDK ou du. AWS CLI

Rubriques

- AWS IoT Core- extrémités du point de terminaison du plan de contrôle
- AWS IoT Core - Points de terminaison du plan de données
- AWS IoT Core- points de terminaison du fournisseur d'informations d'identification
- Points de terminaison de données sur les tâches AWS IoT Device Management -
- AWS IoT Device Management - Points de terminaison Fleet Hub
- AWS IoT Device Management - points de terminaison sécurisés pour le tunneling
- AWS IoT Device Management - Points de terminaison d'intégrations gérés

AWS IoT Core- extrémités du point de terminaison du plan de contrôle

Les points de terminaison du AWS IoT Core - plan de contrôle conformes à la norme FIPS qui prennent en charge les opérations AWS IoT et les commandes CLI associées sont répertoriés dans la section Points de terminaison FIPS par le service. Dans Points de terminaison FIPS par service, recherchez le service AWS IoT Core - plan de contrôle et recherchez le point de terminaison de votre Région AWS.

Pour utiliser le point de terminaison conforme à la norme FIPS lorsque vous accédez aux AWS IoTopérations, utilisez le AWS SDK ou l'API REST avec le point de terminaison qui vous convient. Région AWS

Pour utiliser le point de terminaison conforme à la norme FIPS lorsque vous exécutez des aws iot commandes CLI, ajoutez le paramètre --endpoint avec le point de terminaison approprié pour votre Région AWS à la commande.

AWS IoT Core - Points de terminaison du plan de données

Les points de terminaison du AWS IoT Core - plan de données conformes à la norme FIPS sont répertoriés dans Points de terminaison FIPS par service . Dans Points de terminaison FIPS par service, recherchez le AWS IoT Core du plan de données et recherchez le point de terminaison de votre Région AWS.

Vous pouvez utiliser le point de terminaison conforme à la norme FIPS pour un client conforme à la norme FIPS en utilisant le SDK du AWS IoT périphérique et en fournissant le point de terminaison à la fonction de connexion du SDK à la place du point de terminaison par défaut de votre compte, à savoir le plan de données AWS IoT Core. Région AWS La fonction de connexion est spécifique au SDK du AWS IoT périphérique. Pour un exemple de fonction de connexion, consultez la fonction de connexion dans le SDK AWS IoT Device pour Python.

> **Note**
>
> AWS IoT ne prend pas en charge les points de terminaison du plan de données Compte AWS spécifiques AWS IoT Core qui sont conformes à la norme FIPS. Les fonctionnalités de service qui nécessitent un point de terminaison Compte AWS spécifique dans l'indication du nom du serveur (SNI) ne peuvent pas être utilisées. Conforme à la norme FIPS AWS IoT Core - les points de terminaison du plan de données ne peuvent pas prendre en charge les certificats d'enregistrement multi-comptes, les domaines personnalisés, les autorisations personnalisées et les points de terminaison configurables (y compris les politiques TLS prises en charge).

AWS IoT Core- points de terminaison du fournisseur d'informations d'identification

Les points de terminaison des fournisseurs d'informations d'identification conformes AWS IoT Coreà la norme FIPS sont répertoriés dans la section Points de terminaison FIPS par service. Dans FIPS Endpoints by Service, recherchez le service du AWS IoT Core fournisseur d'informations d'identification et recherchez le point de terminaison correspondant à votre. Région AWS

> **Note**
>
> AWS IoT ne prend pas en charge les points de terminaison des fournisseurs d'informations d'identification Compte AWS spécifiques AWS IoT Core qui sont conformes à la norme FIPS. Les fonctionnalités de service qui nécessitent un point de terminaison Compte AWS spécifique dans l'indication du nom du serveur (SNI) ne peuvent pas être utilisées. Conforme à la norme FIPS AWS IoT Core: les points de terminaison des fournisseurs d'informations d'identification ne peuvent pas prendre en charge les certificats d'enregistrement multi-comptes, les domaines personnalisés, les autorisateurs personnalisés et les points de terminaison configurables (y compris les politiques TLS prises en charge).

Points de terminaison de données sur les tâches AWS IoT Device Management -

Les points de terminaison de données de tâches AWS IoT Device Management - conformes à la norme FIPS sont répertoriés dans Points de terminaison FIPS par service. Dans Points de terminaison FIPS par service, recherchez le service de donnéesAWS IoT Device Management - de tâches et le point de terminaison de votre Région AWS.

Pour utiliser le point de terminaison de donnéesAWS IoT Device Management - de tâches compatible FIPS lorsque vous exécutez des commandes CLIaws iot-jobs-data, ajoutez le paramètre avec le --endpoint point de terminaison approprié pour vous à la commande Région AWS . Vous pouvez également utiliser l'API REST avec ce point de terminaison.

Vous pouvez utiliser le point de terminaison conforme à la norme FIPS pour un client conforme à la norme FIPS en utilisant le SDK pour AWS IoT appareils et en fournissant le point de terminaison à la fonction de connexion du SDK à la place du point de terminaison de données par défaut AWS IoT Device Management de votre compte, à savoir les tâches. Région AWS La fonction de connexion est

spécifique au SDK de l'appareil AWS IoT . Pour un exemple de fonction de connexion, consultez la fonction de connexion dans le SDK AWS IoT Device pour Python.

AWS IoT Device Management - Points de terminaison Fleet Hub

Conforme à la norme FIPS AWS IoT Device Management: les points de terminaison Fleet Hub à utiliser avec les commandes CLI de Fleet Hub pour la gestion des AWS IoT appareils sont répertoriés dans la section Points de terminaison FIPS par service. Dans Points de terminaison FIPS par service, recherchez le service AWS IoT Device Management - Fleet Hub et le point de terminaison de votre Région AWS.

Pour utiliser le point de terminaison Fleet Hub conforme AWS IoT Device Managementà la norme FIPS lorsque vous exécutez des commandes aws iotfleethub CLI, ajoutez le --endpoint paramètre avec le point de terminaison approprié pour vous Région AWS à la commande. Vous pouvez également utiliser l'API REST avec ce point de terminaison.

AWS IoT Device Management - points de terminaison sécurisés pour le tunneling

Les points de terminaison de AWS IoT Device Management - tunneling sécurisé conformes à la AWS IoT norme FIPS pour l'API de tunneling sécurisé et les commandes CLI correspondantes sont répertoriés dans Points de terminaison FIPS par le service . Dans Points de terminaison FIPS par le service, recherchez le service de tunneling sécuriséAWS IoT Device Management - et le point de terminaison de votre Région AWS.

Pour utiliser le AWS IoT Device Management - point de terminaison de tunneling sécurisé conforme à la norme FIPS lorsque vous exécutez des aws iotsecuretunneling commandes CLI, ajoutez le paramètre --endpoint avec le point de terminaison approprié pour votre Région AWS à la commande. Vous pouvez également utiliser l'API REST avec ce point de terminaison.

AWS IoT Device Management - Points de terminaison d'intégrations gérés

Les points de terminaison du plan de contrôle conformes à la norme FIPS qui prennent en charge les opérations d'intégration gérées et leurs AWS CLI commandes associées sont répertoriés dans la section Points de terminaison FIPS par service. Dans FIPS Endpoints by Service, recherchez le service AWS IoT Device Management - Intégrations gérées et recherchez le point de terminaison correspondant à votre. Région AWS

Pour utiliser le point de terminaison conforme à la norme FIPS lorsque vous accédez aux opérations d'intégration gérées, utilisez le AWS SDK ou l'API REST avec le point de terminaison qui vous convient. Région AWS

Pour utiliser le point de terminaison conforme à la norme FIPS lorsque vous exécutez des commandes CLI d'intégrations gérées, ajoutez le --endpoint paramètre avec le point de terminaison approprié pour vous Région AWS à la commande.

Gestion des appareils avec AWS IoT

AWS IoT fournit un registre qui vous aide à gérer les choses. Un objet est une représentation d'un appareil spécifique ou d'une entité logique. Il peut s'agir d'un appareil physique ou d'un capteur (par exemple, une ampoule ou un interrupteur sur un mur). Il peut également s'agir d'une entité logique telle qu'une instance d'une application ou d'une entité physique qui ne se connecte pas à d'autres appareils AWS IoT mais qui est associée à d'autres appareils qui le font (par exemple, une voiture équipée de capteurs de moteur ou d'un panneau de commande).

Les informations concernant un objet sont stockées dans le registre en tant que données JSON. Voici un exemple d'objet :

```
{
     "version": 3,
    "thingName": "MyLightBulb",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

Les objets sont identifiés par un nom. Ils peuvent également avoir des attributs, qui sont des paires nom-valeur, que vous pouvez utiliser pour stocker des informations concernant l'objet, comme son numéro de série ou le fabricant.

Un cas d'utilisation de appareil classique consisterait à utiliser le nom d'un objet en tant qu'ID de client MQTT par défaut. Bien que nous n'imposons pas de mappage entre le nom de registre d'un objet et son utilisation du client MQTT IDs, des certificats ou de l'état fantôme, nous vous recommandons de choisir un nom d'objet et de l'utiliser comme identifiant du client MQTT pour le registre et le service Device Shadow. Cela permet d'organiser votre parc IoT plus facilement, sans perdre la souplesse du modèle de certificat d'appareil ou du shadow sous-jacent.

Vous n'avez pas besoin de créer d'objet dans le registre pour connecter un appareil à AWS IoT. L'ajout d'objets au registre permet de les gérer et de rechercher des appareils plus facilement.

Gérer les choses avec le registre

Vous utilisez la AWS IoT console, AWS IoT l'API ou le AWS CLI pour interagir avec le registre. Les sections suivantes montrent comment utiliser l'interface de ligne de commande pour qu'elle fonctionne avec le registre.

Lorsque vous nommez vos objets objets :

- N'utilisez pas d'informations personnellement identifiables dans le nom de votre objet. Le nom de l'objet peut apparaître dans les communications et les rapports non chiffrés.

Rubriques

- Créer un objet
- Liste des objets
- Décrivez des objets
- Mettre à jour un objet
- Supprimer un objet
- Attacher un mandataire à un objet
- Énumérer les éléments associés à un directeur
- Répertorier les principes associés à un objet
- Lister les éléments associés à une V2 principale
- Répertorier les principes associés à un objet V2
- Détacher un mandataire d'un objet

Créer un objet

La commande suivante montre comment utiliser la AWS IoT CreateThing commande de la CLI pour créer un objet. Vous ne pouvez pas changer le nom d'un objet une fois qu'il est créé. Pour modifier le nom d'un objet, créez-en un nouveau, donnez-lui le nouveau nom, puis supprimez l'ancien.

```
$ aws iot create-thing \
    --thing-type-name "MyLightBulb" \
    --attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

La commande CreateThing affiche le nom et l'Amazon Resource Name (ARN) de votre nouvel objet :

```
{
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678"
}
```

> **Note**
>
> Nous vous déconseillons d'utiliser des informations personnelles identifiables dans le nom de vos objets.

Pour plus d'informations, consultez create-thing dans la AWS CLI Référence des commandes.

Liste des objets

Vous pouvez utiliser la commande ListThings pour répertorier tous les objets présents dans votre compte :

```
$ aws iot list-things
```

```
{
    "things": [
       {
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyLightBulb"
        },
        {
            "attributes": {
                "numOfStates":"3"
             },
            "version": 11,
            "thingName": "MyWallSwitch"
        }
    ]
}
```

Vous pouvez utiliser ListThings la commande pour chercher tous les objets d'un type d'objet specifique :

```
$  aws iot list-things --thing-type-name "LightBulb"
```

```
{
    "things": [
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

Vous pouvez utiliser la ListThings commande pour rechercher tous les objets qui ont un attribut avec une valeur spécifique : Cette commande recherche jusqu'à trois attributs.

```
$  aws iot list-things --attribute-name "wattage" --attribute-value "75"
```

```
{
    "things": [
        {
            "thingTypeName": "StopLight",
            "attributes": {
                "model": "123",
                "wattage": "75"
```

```
            },
            "version": 3,
            "thingName": "MyLightBulb"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

Pour plus d'informations, veuillez consulter list-objects dans la AWS CLI Référence des commandes .

Décrivez des objets

Pour afficher des informations sur un objet, vous pouvez utiliser la commande DescribeThing :

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "version": 3,
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "StopLight",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
```

```
}
```

Pour plus d'informations, voir describe-thing dans la référence des AWS CLI commandes.

Mettre à jour un objet

Vous pouvez utiliser la commande UpdateThing pour mettre à jour un objet : Cette commande met à jour uniquement les attributs de l'objet. Vous ne pouvez pas changer le nom d'un objet. Pour modifier le nom d'un objet, créez-en un nouveau, donnez-lui le nouveau nom, puis supprimez l'ancien.

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":
 {\"wattage\":\"150\", \"model\":\"456\"}}"
```

La commande UpdateThing ne génère pas de sortie. Vous pouvez voir le résultat à l'aide de la commande DescribeThing :

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "attributes": {
        "model": "456",
        "wattage": "150"
    },
    "version": 2,
    "thingName": "MyLightBulb"
}
```

Pour plus d'informations, consultez update-thing dans la AWS CLI Référence de commande.

Supprimer un objet

Vous pouvez utiliser la commande DeleteThing pour supprimer un objet :

```
$ aws iot delete-thing --thing-name "MyThing"
```

Cette commande renvoie un message de succès de l'opération sans erreur si la suppression a été réussie ou que vous spécifiez un objet qui n'existe pas.

Pour plus d'informations, veuillez consulter delete-thing dans la AWS CLI Référence des commandes.

Attacher un mandataire à un objet

Un appareil physique peut utiliser un principal pour communiquer avec AWS IoT. Le principal peut être un certificat X.509 ou un identifiant Amazon Cognito. Vous pouvez associer un certificat ou un identifiant Amazon Cognito à l'élément du registre qui représente votre appareil en exécutant la attach-thing-principalcommande.

Pour associer un certificat ou un identifiant Amazon Cognito à votre objet, utilisez la attach-thing-principalcommande suivante :

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb1" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Pour joindre un certificat à votre objet avec un type de pièce jointe (pièce jointe exclusive ou pièce jointe non exclusive), utilisez la attach-thing-principalcommande et spécifiez un type dans le `--thing-principal-type` champ. Une pièce jointe exclusive signifie que votre objet IoT est le seul élément attaché au certificat, et que ce certificat ne peut être associé à aucun autre élément. Une pièce jointe non exclusive signifie que votre objet IoT est associé au certificat et que ce certificat peut être associé à d'autres éléments. Pour de plus amples informations, veuillez consulter ???.

> **Note**
>
> Pour ??? cette fonctionnalité, vous ne pouvez utiliser le certificat X.509 qu'en tant que principal.

```
$ aws iot attach-thing-principal \
    --thing-name "MyLightBulb2" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
    --thing-principal-type "EXCLUSIVE_THING"
```

Si la pièce jointe est réussie, la AttachThingPrincipal commande ne produit aucune sortie. Pour décrire la pièce jointe, utilisez la commande list-thing-principals-v 2 CLI.

Pour plus d'informations, consultez la référence AttachThingPrincipalde AWS IoT Core l'API.

Énumérer les éléments associés à un directeur

Pour répertorier les éléments associés au principal spécifié, exécutez la `list-principal-things`commande. Notez que cette commande ne répertorie pas le type de pièce jointe entre l'objet et le certificat. Pour répertorier le type de pièce jointe, utilisez la `list-principal-things-v2`commande. Pour de plus amples informations, veuillez consulter ???.

```
$ aws iot list-principal-things \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

Le résultat peut ressembler à ce qui suit.

```
{
    "things": [
        "MyLightBulb1",
        "MyLightBulb2"
    ]
}
```

Pour plus d'informations, consultez la référence ListPrincipalThingsde AWS IoT Core l'API.

Répertorier les principes associés à un objet

Pour répertorier les principes associés à l'objet spécifié, exécutez la `list-thing-principals`commande. Notez que cette commande ne répertorie pas le type de pièce jointe entre l'objet et le certificat. Pour répertorier le type de pièce jointe, utilisez la `list-thing-principals-v2`commande. Pour de plus amples informations, veuillez consulter ???.

```
$ aws iot list-thing-principals \
    --thing-name "MyLightBulb1"
```

Le résultat peut ressembler à ce qui suit.

```
{
```

```
    "principals": [
        "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8",
        "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
    ]
}
```

Pour plus d'informations, consultez la référence ListThingPrincipalsde AWS IoT Core l'API.

Lister les éléments associés à une V2 principale

Pour répertorier les éléments associés au certificat spécifié, ainsi que le type de pièce jointe, exécutez la `list-principal-things-v2`commande. Le type de pièce jointe fait référence à la manière dont le certificat est attaché à l'objet.

```
$ aws iot list-principal-things-v2 \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

Le résultat peut ressembler à ce qui suit.

```
{
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_2"
        }

    ]
}
```

Pour plus d'informations, reportez-vous à la section ListPrincipalThingsV2 de la référence AWS IoT Core d'API.

Répertorier les principes associés à un objet V2

Pour répertorier les certificats associés à l'objet spécifié, ainsi que le type de pièce jointe, exécutez la `list-thing-principals-V2`commande. Le type de pièce jointe fait référence à la manière dont le certificat est attaché à l'objet.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

Le résultat peut ressembler à ce qui suit.

```
{
    "ThingPrincipalObjects": [
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        },
        {
            "thingPrincipalType": "NON_EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
1a234b39b4b68278f2e9d84bf97eac2cbf4a1c28b23ea29a44559b9bcf8d395b"
        }
    ]
}
```

Pour plus d'informations, reportez-vous à la section ListThingsPrincipalV2 de la référence AWS IoT Core d'API.

Détacher un mandataire d'un objet

Vous pouvez utiliser la commande `DetachThingPrincipal` pour détacher un certificat d'un objet :

```
$ aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
```

La DetachThingPrincipal commande ne génère pas de sortie.

Pour plus d'informations, consultez la référence detach-thing-principalde AWS IoT Core l'API.

Types d'objets

Les types d'objets permettent de stocker des informations de configuration et de description communes à tous les objets qui associés au même type d'objet. Cela simplifie la gestion des objets dans le registre. Par exemple, vous pouvez définir un type d' LightBulb objet. Tous les objets associés au type d' LightBulb objet partagent un ensemble d'attributs : numéro de série, fabricant et puissance en watts. Lorsque vous créez un objet de type LightBulb (ou que vous modifiez le type d'un objet existant en LightBulb), vous pouvez spécifier des valeurs pour chacun des attributs définis dans le type d' LightBulb objet.

Bien que les types d'objet soient facultatifs, leur utilisation facilite la découverte des objets.

- Les objets avec un type d'objet peuvent posséder jusqu'à 50 attributs.
- Les objets sans type d'objet peuvent posséder jusqu'à trois attributs.
- Un objet ne peut être associé qu'à un seul type d'objet.
- Il n'y a aucune limite au nombre de types d'objets que vous pouvez créer dans votre compte.

Vous ne pouvez pas modifier un nom de type d'objet après sa création. Vous pouvez rendre obsolète un type d'objet à tout moment pour empêcher de nouveaux objets d'y être associés. Vous pouvez aussi supprimer les types d'objets qui n'ont aucun objet associé.

Rubriques :

- Créer un type d'objet
- Liste des types d'objets
- Décrire un type d'objet
- Associer un type d'objet à un objet
- Mettre à jour un type d'objet
- Rendre obsolète un type d'objet
- Supprimer un type d'objet

Créer un type d'objet

Vous pouvez utiliser la commande CreateThingType pour créer un type d'objet :

```
$ aws iot create-thing-type

               --thing-type-name "LightBulb" --thing-type-properties
 "thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

La commande CreateThingType renvoie une réponse qui contient le type d'objet et son ARN :

```
{
    "thingTypeName": "LightBulb",
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
}
```

Liste des types d'objets

Vous pouvez utiliser la commande ListThingTypes pour répertorier les types d'objets :

```
$ aws iot list-thing-types
```

La ListThingTypes commande renvoie une liste des types d'objets définis dans votre Compte AWS :

```
{
    "thingTypes": [
        {
            "thingTypeName": "LightBulb",
            "thingTypeProperties": {
                "searchableAttributes": [
                    "wattage",
                    "model"
                ],
                "thingTypeDescription": "light bulb type"
            },
            "thingTypeMetadata": {
                "deprecated": false,
                "creationDate": 1468423800950
            }
```

```
        }
    ]
}
```

Décrire un type d'objet

Vous pouvez utiliser la commande DescribeThingType pour obtenir des informations sur un type d'objet :

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

La commande DescribeThingType renvoie des informations sur le type spécifié :

```
{
    "thingTypeProperties": {
        "searchableAttributes": [
            "model",
            "wattage"
        ],
        "thingTypeDescription": "light bulb type"
    },
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
    "thingTypeName": "LightBulb",
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1544466338.399
    }
}
```

Associer un type d'objet à un objet

Vous pouvez utiliser la commande CreateThing pour spécifier un type d'objet lorsque vous créez un objet :

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Vous pouvez utiliser la commande UpdateThing à tout moment pour modifier le type d'objet associé à un objet :

```
$ aws iot update-thing --thing-name "MyLightBulb"
                --thing-type-name "LightBulb" --attribute-payload  "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Vous pouvez également utiliser la commande UpdateThing pour dissocier un objet d'un type d'objet.

Mettre à jour un type d'objet

Vous pouvez utiliser la UpdateThingType commande pour mettre à jour un type d'objet lorsque vous créez un objet :

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Vous pouvez utiliser la commande UpdateThing à tout moment pour modifier le type d'objet associé à un objet :

```
$ aws iot update-thing --thing-name "MyLightBulb"
                --thing-type-name "LightBulb" --attribute-payload  "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

Vous pouvez également utiliser la commande UpdateThing pour dissocier un objet d'un type d'objet.

Rendre obsolète un type d'objet

Les types d'objets sont immuables. Il est impossible de les modifier une fois qu'ils sont définis. Vous pouvez, toutefois, rendre obsolète un type d'objet pour empêcher les utilisateurs de lui associer de nouveaux objets. Tous les objets existants associés au type d'objet restent inchangés.

Pour rendre obsolète un type d'objet, utilisez la commande DeprecateThingType :

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

Vous pouvez voir le résultat à l'aide de la commande DescribeThingType :

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
```

```
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type",
    },
    "thingTypeMetadata": {
        "deprecated": true,
        "creationDate": 1468425854308,
        "deprecationDate": 1468446026349
    }
}
```

Rendre obsolète un type d'objet est une opération réversible. Vous pouvez annuler une obsolescence en utilisant l'indicateur `--undo-deprecate` avec la commande CLI DeprecateThingType :

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

Vous pouvez voir le résultat à l'aide de la commande CLI DescribeThingType :

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/StopLight",
    "thingTypeId": "12345678abcdefgh12345678ijklmnop12345678"
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type"
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1468425854308,
    }
}
```

Supprimer un type d'objet

Vous pouvez supprimer des types d'objet uniquement une fois qu'ils sont obsolètes. Pour supprimer un type d'objet, utilisez la commande DeleteThingType :

```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

> **Note**
>
> Avant de pouvoir supprimer un type d'objet, attendez cinq minutes après l'avoir rendu obsolète.

Groupes d’objets statiques

Les groupes d'objets permettent de gérer plusieurs objets simultanément en les classant dans des groupes. Les groupes d’objets statiques contiennent des objets gérés grâce à la console, l'interface de ligne de commande ou l'API. Les groupes d’objets dynamiques, en revanche, contiennent des objets qui correspondent à une requête spécifiée. Les groupes d’objets statiques peuvent également contenir d'autres groupes d’objets statiques — vous pouvez créer une hiérarchie de groupes. Vous pouvez attacher à un groupe parent une stratégie dont héritent tous ses groupes enfants et tous les objets du groupe, ainsi que leurs groupes enfants. Cela facilite le contrôle des autorisations pour les grands nombres d'objets.

> **Note**
>
> Les politiques relatives aux groupes d'objets n'autorisent pas l'accès aux opérations du plan de AWS IoT Greengrass données. Pour autoriser un objet à accéder à une opération de plan de AWS IoT Greengrass données, ajoutez l'autorisation à une AWS IoT politique que vous associez au certificat de l'objet. Pour de plus amples informations, veuillez consulter Authentification et autorisation de l'appareil dans le AWS IoT Greengrass guide du développeur.

Voici ce que vous pouvez faire avec les groupes d'objets statiques :

- Créer, décrire ou supprimer un groupe.

- Ajouter un objet à un ou plusieurs groupes.
- Supprimer un objet d'un groupe.
- Répertorier les groupes que vous avez créés.
- Répertorier tous les groupes enfants d'un groupe (ses descendants directs et indirects).
- Répertorier les objets d'un groupe, y compris tous les objets de ses groupes enfants.
- Répertorier tous les groupes ascendants d'un groupe (ses parents directs et indirects).
- Ajouter, supprimer ou mettre à jour les attributs d'un groupe. Les attributs sont des paires nom-valeur que vous pouvez utiliser afin de stocker des informations relatives à un groupe.
- Attacher une stratégie à un groupe ou la détacher de celui-ci.
- Répertorier les stratégies attachées à un groupe.
- Répertorier les stratégies dont un objet hérite (en fonction des stratégies attachées à son groupe ou à l'un de ses groupes parents).
- Configurer les options de journalisation des objets d'un groupe. Voir Configuration de la AWS IoT journalisation.
- Créer des tâches qui sont envoyées vers et exécutées sur chaque objet d'un groupe et de ses groupes enfants. Voir AWS IoT Emplois.

> **Note**
>
> Lorsqu'un objet est attaché à un groupe d'objets statique auquel une AWS IoT Core politique est attachée, le nom de l'objet doit correspondre à l'ID du client.

Voici quelques restrictions relatives aux groupes d'objets statiques :

- Un groupe peut avoir un parent direct au maximum.
- Si un groupe est un enfant d'un autre groupe, indiquez-le au moment de sa création.
- Vous ne pouvez pas modifier le parent d'un groupe ultérieurement. Veillez donc à planifier votre hiérarchie de groupe et à créer un groupe parent avant de créer les groupes enfants qu'il contient.
- Le nombre de groupes auxquels un objet peut appartenir est limité.
- Vous ne pouvez pas ajouter un objet à plusieurs groupes de la même hiérarchie. (En d'autres termes, vous ne pouvez pas ajouter un objet à deux groupes qui partagent un même parent).
- Vous ne pouvez pas renommer un groupe.

- Les noms des groupes d'objets ne peuvent contenir aucun caractère international, comme û, é et ñ.
- N'utilisez pas d'informations personnellement identifiables dans le nom de votre groupe d'objets. Le nom de groupe d'objet peut apparaître dans les communications et les rapports non chiffrés.

Le fait d'attacher des stratégies aux groupes ou de les détacher de ceux-ci vous permet d'améliorer la sécurité des opérations AWS IoT de nombreuses façons importantes. La méthode par appareil qui consiste à attacher une stratégie à un certificat, qui est lui-même attaché ensuite à un objet, prend du temps et complique la mise à jour ou la modification rapide des stratégies pour tout un parc d'appareils. Le fait d'attacher une stratégie au groupe de l'objet permet d'éviter certaines étapes lors de la rotation des certificats pour un objet. De plus, les stratégies sont appliquées dynamiquement aux objets lorsqu'elles changent d'appartenance à un groupe, ce qui signifie que vous n'avez pas besoin de recréer un ensemble complexe d'autorisations chaque fois qu'un appareil change d'appartenance dans un groupe.

Créer un groupe d’objets statiques

Utilisez la commande CreateThingGroup pour créer un groupe d'objets statiques.

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

La commande CreateThingGroup renvoie une réponse qui contient le nom, l'ID et l'ARN du groupe d’objets statiques :

```
{
    "thingGroupName": "LightBulbs",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

> **Note**
>
> Nous vous déconseillons d'utiliser des informations personnelles identifiables dans le nom de vos groupes d'objets.

Voici un exemple qui spécifie un parent du groupe d'objets statiques lors de sa création :

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name
 LightBulbs
```

Comme auparavant, la commande CreateThingGroup renvoie une réponse qui contient le nom, l'ID et l'ARN du groupe d'objets statiques :

```
{
    "thingGroupName": "RedLights",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

Important

Gardez à l'esprit les limites suivantes lorsque vous créez des hiérarchies de groupes d'objets :

- Un groupe d'objets ne peut avoir qu'un seul parent direct.
- Le nombre de groupes d'enfants directs qu'un groupe d'objets peut avoir est limité.
- Le nombre maximal de niveaux d'une hiérarchie de groupes d'objets est limité.
- Le nombre d'attributs qu'un groupe d'objets peut avoir est limité. Les attributs sont des paires nom-valeur que vous pouvez utiliser afin de stocker des informations relatives à un groupe. La longueur du nom de chaque attribut et de chaque valeur est également limitée.

Décrire un groupe d'objets

Pour obtenir des informations sur un groupe d'objets, vous pouvez utiliser la commande DescribeThingGroup :

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

La commande DescribeThingGroup renvoie des informations sur le groupe spécifié :

```
{
    "thingGroupName": "RedLights",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
```

```
    "thingGroupId": "12345678abcdefgh12345678ijklmnop12345678",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1478299948.882
        "parentGroupName": "Lights",
        "rootToParentThingGroups": [
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ShinyObjects",
                "groupName": "ShinyObjects"
            },
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs",
                "groupName": "LightBulbs"
            }
        ]
    },
    "thingGroupProperties": {
        "attributePayload": {
            "attributes": {
                "brightness": "3400_lumens"
            },
        },
        "thingGroupDescription": "string"
    },
}
```

Ajouter un objet à un groupe d’objets statiques

Vous pouvez utiliser la commande AddThingToThingGroup pour ajouter un objet à un groupe d’objets statiques :

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

La commande AddThingToThingGroup ne génère pas de sortie.

> **Important**
>
> Vous pouvez ajouter un objet à un maximum de 10 groupes. Mais vous ne pouvez pas ajouter un objet à plusieurs groupes de la même hiérarchie. (En d'autres termes, vous ne pouvez pas ajouter un objet à deux groupes qui partagent un même parent.)

Si un objet appartient à autant de groupes d'objets que possible et qu'un ou plusieurs de ces groupes est un groupe d'objets dynamiques, vous pouvez utiliser l'indicateur overrideDynamicGroups pour que les groupes statiques soient prioritaires par rapport aux groupes dynamiques.

Supprimer un objet d'un groupe d’objet statiques

Vous pouvez utiliser la commande RemoveThingFromThingGroup pour supprimer un objet d'un groupe :

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

La commande RemoveThingFromThingGroup ne génère pas de sortie.

Répertorier les objets d'un groupe d'objets

Vous pouvez utiliser la commande ListThingsInThingGroup pour répertorier les objets appartenant à un groupe :

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

La commande ListThingsInThingGroup renvoie une liste des objets d'un groupe donné :

```
{
    "things":[
        "TestThingA"
    ]
}
```

Le paramètre `--recursive` vous permet de répertorier les objets appartenant à un groupe et ceux figurant dans ses groupes enfants :

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
    "things":[
```

```
        "TestThingA",
        "MyLightBulb"
    ]
}
```

> **Note**
>
> Cette opération est cohérente à terme. En d'autres termes, les modifications apportées au groupe d'objets peuvent ne pas être reflétées immédiatement.

Répertorier les groupes d'objets

Vous pouvez utiliser la commande ListThingGroups pour répertorier les groupes d’objets de votre compte :

```
$ aws iot list-thing-groups
```

La ListThingGroups commande renvoie une liste des groupes d'objets de votre Compte AWS :

```
{
    "thingGroups": [
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
        {
            "groupName": "ReplaceableObjects",
```

```
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
    ]
}
```

Utilisez les filtres facultatifs pour répertorier les groupes ayant un groupe donné comme parent (`--parent-group`) ou ceux dont le nom commence par un préfixe spécifique (`--name-prefix-filter`). Le paramètre `--recursive` vous permet de répertorier tous les groupes enfants, et pas seulement les groupes enfants directs d'un groupe d'objets :

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

Dans ce cas, la ListThingGroups commande renvoie une liste des groupes enfants directs du groupe d'objets défini dans votre Compte AWS :

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        }
    ]
}
```

Utilisez le paramètre `--recursive` avec la commande ListThingGroups pour répertorier tous les groupes enfants d'un groupe d'objets, et pas seulement les enfants directs :

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

La commande ListThingGroups renvoie une liste de tous les groupes enfants du groupe d'objets :

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
```

```
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
    ]
}
```

> **Note**
>
> Cette opération est cohérente à terme. En d'autres termes, les modifications apportées au groupe d'objets peuvent ne pas être reflétées immédiatement.

Répertorier les groupes d'un objet

Vous pouvez utiliser la commande ListThingGroupsForThing pour répertorier les objets appartenant à un groupe :

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

La ListThingGroupsForThing commande renvoie une liste des groupes d'objets auxquels appartient un objet :

```
{
    "thingGroups":[
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
ReplaceableObjects"
        }
```

```
    ]
}
```

Mettre à jour un groupe d'objets statiques

Vous pouvez utiliser la commande UpdateThingGroup afin de mettre à jour les attributs d'un groupe d'objets statiques :

```
$ aws iot update-thing-group --thing-group-name "LightBulbs" --thing-group-properties
 "thingGroupDescription=\"this is a test group\", attributePayload=\"{\"attributes
\"={\"Owner\"=\"150\",\"modelNames\"=\"456\"}}"
```

La commande UpdateThingGroup renvoie une réponse qui contient le numéro de version du groupe après la mise à jour :

```
{
    "version": 4
}
```

> **Note**
>
> Le nombre d'attributs qu'un objet peut avoir est limité.

Supprimer un groupe d'objets

Pour supprimer un groupe d'objets, utilisez la commande DeleteThingGroup :

```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

La commande DeleteThingGroup ne génère pas de sortie.

> **Important**
>
> Si vous essayez de supprimer un groupe d'objets qui comporte des groupes d'objets enfants, vous recevez une erreur :
>
> ```
> A client error (InvalidRequestException) occurred when calling the
>  DeleteThingGroup
> ```

```
operation: Cannot delete thing group : RedLights when there are still child
 groups attached to it.
```

Avant de supprimer le groupe, supprimez d'abord tous les groupes d'enfants.

Vous pouvez supprimer un groupe qui a des objets enfants, mais les autorisations accordées aux objets par appartenance au groupe ne s'appliqueront plus. Avant de supprimer un groupe auquel une stratégie est attachée, vérifiez que la suppression de ces autorisations n'empêchera pas les objets du groupe de fonctionner correctement. En outre, les commandes qui indiquent à quels groupes appartient un objet (par exempleListGroupsForThing) peuvent continuer à afficher le groupe pendant la mise à jour des enregistrements dans le cloud.

Attacher une stratégie à un groupe d’objets statiques

Vous pouvez utiliser la commande AttachPolicy pour attacher une stratégie à un groupe d'objets statiques et, par extension, à tous les objets de ce groupe et de ses groupes enfants :

```
$ aws iot attach-policy \
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \
  --policy-name "myLightBulbPolicy"
```

La commande AttachPolicy ne génère pas de sortie

Important

Vous pouvez attacher au maximum deux stratégies à un groupe.

Note

Nous vous déconseillons d'utiliser des informations personnelles identifiables dans le nom de votre politique.

Le paramètre `--target` peut être un ARN de groupe d'objets (comme ci-dessus), un certificat d’ARN ou une identité Amazon Cognito. Pour plus d'informations sur les stratégies, les certificats et l'authentification, consultez Authentification.

Pour plus d'informations, consultez AWS IoT Core Stratégies .

Détacher une stratégie d'un groupe d’objets statiques

Vous pouvez utiliser la commande DetachPolicy pour détacher une stratégie d'un groupe d'objets et, par extension, de tous les objets de ce groupe et de ses groupes enfants :

```
$ aws iot detach-policy --target "arn:aws:iot:us-west-2:123456789012:thinggroup/
LightBulbs" --policy-name "myLightBulbPolicy"
```

La commande DetachPolicy ne génère pas de sortie.

Répertorier les stratégies attachées à un groupe d'objets statiques

Vous pouvez utiliser la commande ListAttachedPolicies pour répertorier les stratégies attachées à un groupe d’objets statiques :

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-
west-2:123456789012:thinggroup/RedLights"
```

Le `--target` paramètre peut être un ARN de groupe d'objets (comme ci-dessus), un ARN de certificat ou une identité Amazon Cognito.

Ajoutez le paramètre facultatif `--recursive` afin d'inclure toutes les stratégies attachées aux groupes parents du groupe.

La commande ListAttachedPolicies renvoie une liste de stratégies :

```
{
    "policies": [
        "MyLightBulbPolicy"
        ...
    ]
}
```

Répertorier les groupes d'une stratégie

Vous pouvez utiliser la commande ListTargetsForPolicy afin de répertorier les cibles, y compris tous les groupes éventuels, auxquelles une stratégie est attachée :

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

Ajoutez le paramètre facultatif `--page-size` *`number`* afin de spécifier le nombre maximal de résultats renvoyés par chaque demande, et le paramètre `--marker` *`string`* sur les appels suivants afin d'extraire l'ensemble de résultats suivant, le cas échéant.

La commande ListTargetsForPolicy renvoie une liste des cibles et des jetons à utiliser pour extraire davantage de résultats :

```
{
    "nextMarker": "string",
    "targets": [ "string" ... ]
}
```

Obtenir des stratégies efficaces pour un objet

Vous pouvez utiliser la commande GetEffectivePolicies afin de répertorier les stratégies en vigueur pour un objet, y compris les stratégies attachées aux groupes auxquels l'objet appartient (que le groupe soit un parent direct ou un ancêtre indirect) :

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

Utilisez le paramètre `--principal` afin de spécifier l'ARN du certificat attaché à l'objet. Si vous avez opté pour l'authentification d'identité Amazon Cognito, utilisez le `--cognito-identity-pool-id` paramètre et ajoutez si nécessaire `--principal` Amazon Cognitole paramètre afin de spécifier une identité . Si vous spécifiez uniquement le `--cognito-identity-pool-id`, les stratégies associées à ce rôle du pool d'identités pour les utilisateurs non authentifiés sont renvoyées. Si vous utilisez les deux, les stratégies associées à ce rôle du pool d'identités pour les utilisateurs authentifiés sont renvoyées.

Le paramètre `--thing-name` est facultatif et peut être utilisé à la place du paramètre `--principal`. Lorsqu'il est utilisé, les stratégies attachées aux groupes auxquels l'objet appartient et les stratégies attachées aux groupes parents de ces groupes (jusqu'au groupe racine dans la hiérarchie) sont renvoyées.

La commande GetEffectivePolicies renvoie une liste de stratégies :

```
{
    "effectivePolicies": [
        {
            "policyArn": "string",
            "policyDocument": "string",
            "policyName": "string"
        }
        ...
    ]
}
```

Tester l'autorisation pour les actions MQTT

Vous pouvez utiliser la TestAuthorization commande afin de tester si une action MQTT (`Publish`, `Subscribe`) est autorisée pour un objet :

```
aws iot test-authorization \
    --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
    --auth-infos "{\"actionType\": \"PUBLISH\", \"resources\": [ \"arn:aws:iot:us-
east-1:123456789012:topic/my/topic\"]}"
```

Utilisez le paramètre `--principal` afin de spécifier l'ARN du certificat attaché à l'objet. Si vous utilisez l'authentification d'identité Amazon Cognito, spécifiez une identité Cognito comme `--principal` ou utilisez le `--cognito-identity-pool-id` paramètre , ou les deux. Si vous spécifiez uniquement le paramètre `--cognito-identity-pool-id`, les stratégies associées à ce rôle du pool d'identités pour les utilisateurs non authentifiés sont appliquées. Si vous utilisez les deux, les stratégies associées à ce rôle du pool d'identités pour les utilisateurs authentifiés sont appliquées.

Spécifiez une ou plusieurs actions MQTT que vous souhaitez tester en répertoriant des ensembles de ressources et de types d'actions après le paramètre `--auth-infos`. Le champ `actionType` doit contenir « PUBLISH », « SUBSCRIBE », « RECEIVE » ou « CONNECT ». Le `resources` champ doit contenir une liste de ressources ARNs. Pour plus d'informations, consultez AWS IoT Core politiques.

Vous pouvez tester les conséquences de l'ajout des stratégies en les spécifiant avec le paramètre `--policy-names-to-add`. Ou vous pouvez tester les conséquences de la suppression des stratégies en les spécifiant avec le paramètre `--policy-names-to-skip`.

Vous pouvez utiliser le paramètre facultatif `--client-id` pour affiner vos résultats.

La commande TestAuthorization renvoie des détails sur les actions qui ont été autorisées ou refusées pour chaque ensemble de requêtes `--auth-infos` que vous avez spécifié :

```
{
    "authResults": [
        {
            "allowed": {
                "policies": [
                    {
                        "policyArn": "string",
                        "policyName": "string"
                    }
                ]
            },
            "authDecision": "string",
            "authInfo": {
                "actionType": "string",
                "resources": [ "string" ]
            },
            "denied": {
                "explicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                },
                "implicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                }
            },
            "missingContextValues": [ "string" ]
        }
    ]
}
```

Groupes d'objets dynamiques

Les groupes d'objets dynamiques sont créés à partir de requêtes de recherche spécifiques dans le registre. Les paramètres de requête de recherche tels que la connectivité des appareils, la création d'ombres sur les appareils et les données relatives aux AWS IoT Device Defender violations le confirment. Les groupes d'objets dynamiques nécessitent que l'indexation du parc soit activée pour indexer, rechercher et agréger les données de vos appareils. Vous pouvez prévisualiser les éléments d'un groupe d'objets dynamique à l'aide d'une requête de recherche d'indexation de flotte avant de le créer. Pour plus d'informations, consultez Indexation de la flotte et Syntaxe de requête.

> **Note**
>
> Les opérations de groupes d'objets dynamiques sont mesurées dans le cadre des opérations de registre. Pour plus d'informations, consultez AWS IoT Core additional metering details.

Voici les raisons qui différencient les groupes d'objets dynamiques des groupes d'objets statiques :

- L'appartenance des objets n'est pas explicitement définie. Pour créer un groupe d'objets dynamique, définissez une chaîne de requête de recherche pour déterminer l'appartenance au groupe.
- Les groupes d'objets dynamiques ne peuvent pas faire partie d'une hiérarchie.
- Les groupes d'objets dynamiques ne peuvent pas avoir de politiques qui leur sont appliquées.
- Vous utilisez un ensemble de commandes différent pour créer, mettre à jour et supprimer des groupes d'objets dynamiques. Pour toutes les autres opérations, vous utilisez les mêmes commandes pour les deux types de groupes d'objets.
- Le nombre de groupes dynamiques par groupe Compte AWS est limité.

- N'utilisez pas d'informations personnellement identifiables dans le nom de votre groupe d'objets. Le nom de groupe d'objet peut apparaître dans les communications et les rapports non chiffrés.

Pour plus d'informations sur les groupes d'objets statiques, consultez Groupes d'objets statiques.

Cas d'utilisation de groupes d'objets dynamiques

Vous pouvez utiliser des groupes d'objets dynamiques dans les cas d'utilisation suivants :

Spécifier un groupe d'objets dynamique comme cible pour une tâche

La création d'une tâche continue avec un groupe d'objets dynamique comme cible vous permet de cibler automatiquement les appareils lorsqu'ils répondent aux critères souhaités. Les critères peuvent être l'état de connectivité ou tout autre critère stocké dans le registre ou dans le shadow, tel que la version ou le modèle du logiciel. Si un objet n'apparaît pas dans le groupe d'objets dynamique, il ne recevra pas le document correspondant au travail.

Par exemple, si votre parc d'appareils nécessite une mise à jour du microprogramme afin de minimiser le risque d'interruption pendant le processus de mise à jour, et que vous souhaitez uniquement mettre à jour le microprogramme sur les appareils dont l'autonomie de la batterie est supérieure à 80 %. Vous pouvez créer un groupe d'objets dynamique appelé 80 PercentBatteryLife qui inclut uniquement les appareils dont l'autonomie de la batterie est supérieure à 80 % et l'utiliser comme cible pour votre travail. Seuls les appareils répondant à vos critères d'autonomie recevront la mise à jour du microprogramme. Lorsque les appareils atteignent le seuil d'autonomie de la batterie de 80 %, ils sont automatiquement ajoutés au groupe d'objets dynamiques et reçoivent la mise à jour du microprogramme.

Vous pouvez également avoir plusieurs modèles d'appareils dotés de microprogrammes ou de systèmes d'exploitation différents, ce qui nécessite différentes versions des nouvelles mises à jour logicielles. Il s'agit du cas d'utilisation le plus courant pour les groupes dynamiques avec des tâches continues, dans lesquels vous pouvez créer un groupe dynamique pour chaque combinaison de modèle d'appareil, de microprogramme et de système d'exploitation. Vous pouvez ensuite configurer des tâches continues pour chacun de ces groupes dynamiques afin de diffuser les mises à jour logicielles au fur et à mesure que les appareils deviennent membres de ces groupes en fonction des critères définis.

Pour de plus amples informations sur la spécification de groupes de choses en tant que cibles de travail, veuillez consulter CreateJob.

Utilisez les modifications dynamiques de l'appartenance à un groupe pour effectuer les actions souhaitées

Chaque fois qu'un appareil est ajouté ou supprimé d'un groupe d'objets dynamiques, une notification est envoyée à un sujet MQTT dans le cadre des mises à jour des événements du registre. Vous pouvez configurer AWS IoT Core des règles pour interagir avec les AWS services en fonction des mises à jour dynamiques de l'appartenance aux groupes et prendre les mesures souhaitées. Les actions incluent par exemple l'écriture Amazon DynamoDB, l'appel d'une fonction Lambda ou l'envoi d'une notification à Amazon SNS.

Ajoutez des appareils à un groupe d'objets dynamique pour une détection automatique des violations

AWS IoT Device Defender Les clients de Detect peuvent définir un profil de sécurité sur un groupe d'objets dynamique. Les appareils du groupe d'objets dynamique sont automatiquement détectés en cas de violation par le profil de sécurité défini dans le groupe.

Définissez des niveaux de journalisation sur des groupes d'objets dynamiques afin d'observer les appareils grâce à une journalisation précise

Vous pouvez spécifier un niveau de journalisation pour un groupe d'objets dynamique. Cela est utile si vous souhaitez uniquement personnaliser le niveau de journalisation et les détails pour les appareils répondant à certains critères. Par exemple, si vous pensez que des appareils dotés d'une certaine version du microprogramme sont à l'origine d'erreurs dans le sujet publié d'une règle spécifique, vous pouvez définir une journalisation détaillée pour résoudre ces problèmes. Dans ce cas, vous pouvez créer un groupe dynamique pour tous les appareils dotés de cette version du microprogramme, qui, nous le supposons, est stockée sous forme d'attribut de registre ou dans un espace virtuel. Vous pouvez ensuite définir un niveau de débogage, la cible de journalisation étant définie comme ce groupe d'objets dynamique. Pour plus d'informations sur la journalisation précise, voir Surveillance à AWS IoT l'aide CloudWatch des journaux. Pour plus d'informations sur la manière de spécifier un niveau de journalisation pour un groupe d'objets spécifique, voir Configurer la connexion spécifique aux ressources. AWS IoT

Créer un groupe d'objets dynamique

Utilisez la commande CreateDynamicThingGroup pour créer un groupe d'objets dynamique. Pour créer un groupe d'objets dynamique pour le PercentBatteryLife scénario 80, utilisez la commande create-dynamic-thing-group CLI :

```
$ aws iot create-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --query-string "attributes.batterylife80"
```

> **Note**
>
> N'utilisez pas d'informations personnellement identifiables dans les noms de vos groupes d'objets dynamiques.

La CreateDynamicThingGroup commande renvoie une réponse. La réponse contient le nom de l'index, la chaîne de requête, la version de la requête, le nom du groupe d'objets, l'ID du groupe d'objets et le nom de ressource Amazon (ARN) de votre groupe d'objets :

```
{
    "indexName": "AWS_Things",
    "queryVersion": "2017-09-30",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
    "queryString": "attributes.batterylife80\n",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx"
}
```

La création de groupes d'objets dynamiques ne se fait pas en une seule fois. Le remplissage complet d'un groupe d'objets dynamique prend un certain temps. Lorsque vous créez un groupe d'objets dynamique, le statut du groupe est défini sur`BUILDING`. Une fois le remplissage terminé, l'état passe à `ACTIVE`. Pour vérifier l'état de votre groupe d'objets dynamique, utilisez la DescribeThingGroupcommande.

Décrire un groupe d'objets dynamique

Utilisez la commande DescribeThingGroup pour obtenir des informations sur un groupe d'objets dynamique :

```
$ aws iot describe-thing-group --thing-group-name "80PercentBatteryLife"
```

La commande DescribeThingGroup renvoie des informations sur le groupe spécifié :

```
{
    "status": "ACTIVE",
    "indexName": "AWS_Things",
    "thingGroupName": "80PercentBatteryLife",
    "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
    "queryString": "attributes.batterylife80\n",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1548716921.289
    },
```

```
    "thingGroupProperties": {},
    "queryVersion": "2017-09-30",
    "thingGroupId": "84dd9b5b-2b98-4c65-84e4-be0e1ecf4fd8"
}
```

L'exécution DescribeThingGroup sur un groupe d'objets dynamique renvoie des attributs spécifiques aux groupes d'objets dynamiques. Les exemples d'attributs de retour sont le QueryString et le status.

Le statut d'un groupe d'objets dynamique peut avoir les valeurs suivantes :

`ACTIVE`

Le groupe d'objets dynamique est prêt à l'emploi.

`BUILDING`

Le groupe d'objets dynamique est en cours de création et l'appartenance des objets est en cours de traitement.

`REBUILDING`

L'appartenance au groupe d'objets dynamique est en cours de mise à jour, en fonction de l'ajustement de la requête de recherche du groupe.

> **Note**
>
> Après avoir créé un groupe d'objets dynamique, utilisez-le quel que soit son statut. Seuls les groupes d'objets dynamiques ayant le statut `ACTIVE` incluent tous les objets qui correspondent à la requête de recherche de ce groupe d'objets dynamique. Les groupes d'objets dynamiques ayant les statuts `BUILDING` et `REBUILDING` peuvent ne pas inclure tous les objets correspondant à la requête de recherche.

Mettre à jour un groupe d'objets dynamique

Utilisez la commande UpdateDynamicThingGroup pour mettre à jour les attributs d'un groupe d'objets dynamique, y compris la requête de recherche du groupe. La commande suivante met à jour deux attributs. L'une est la description du groupe d'objets, et l'autre est la chaîne de requête qui modifie les critères d'adhésion pour indiquer que l'autonomie de la batterie est supérieure à 85 :

```
$ aws iot update-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --thing-
group-properties "thingGroupDescription=\"This thing group contains devices with a
 battery life greater than 85 percent.\"" --query-string "attributes.batterylife85"
```

La commande UpdateDynamicThingGroup renvoie une réponse qui contient le numéro de version du groupe après la mise à jour :

```
{
    "version": 2
}
```

La mise à jour d'un groupe d'objets dynamique ne se produit pas immédiatement. Le remplissage complet d'un groupe d'objets dynamique prend un certain temps. Lorsque vous mettez à jour un groupe d'objets dynamique, le statut du groupe devient `REBUILDING` alors que le groupe met à jour ses membres. Une fois le remplissage terminé, l'état passe à `ACTIVE`. Pour vérifier l'état de votre groupe d'objets dynamique, utilisez la DescribeThingGroupcommande.

Supprimer un groupe d'objets dynamique

Utilisez la commande DeleteDynamicThingGroup pour supprimer un groupe d'objets dynamique :

```
$ aws iot delete-dynamic-thing-group --thing-group-name "80PercentBatteryLife"
```

La DeleteDynamicThingGroup commande ne génère pas de sortie.

Les commandes qui affichent à quels groupes un objet appartient (par exemple, ListGroupsForThing) peuvent continuer à afficher le groupe pendant que les enregistrements sont mis à jour sur le cloud.

Limitations des groupes d'objets dynamiques et statiques

Les groupes d'objets dynamiques et les groupes d'objets statiques présentent les limites suivantes :

- Le nombre d'attributs qu'un groupe d'objets peut avoir est limité.
- Le nombre de groupes auxquels un objet peut appartenir est limité.
- Vous ne pouvez pas renommer les groupes d'objets.
- Les noms des groupes d'objets ne peuvent contenir aucun caractère international, comme û, é et ñ.

Limites relatives aux groupes d'objets dynamiques

Les groupes d'objets dynamiques présentent les limites suivantes :

Indexation de la flotte

Lorsque le service d'indexation de flotte est activé, vous pouvez effectuer des requêtes de recherche sur votre parc d'appareils. Vous pouvez créer et gérer des groupes d'objets dynamiques une fois le remblayage d'indexation de la flotte terminé. Le délai d'exécution du processus de remblayage dépend directement de la taille de votre parc d'appareils enregistré auprès du. AWS Cloud Une fois que vous avez activé le service d'indexation de flotte pour les groupes d'objets dynamiques, vous ne pouvez pas le désactiver tant que vous n'avez pas supprimé tous vos groupes d'objets dynamiques.

> **Note**
>
> Si vous disposez d'autorisations pour interroger l'index de la flotte, vous pouvez accéder aux données d'objets dans la totalité de la flotte.

Le nombre de groupes d’objets dynamiques est limité

Le nombre de groupes d'objets dynamiques est limité.

Les commandes réussies peuvent enregistrer les erreurs

Lorsque vous créez ou mettez à jour un groupe d'objets dynamique, il est possible que certains éléments soient éligibles à l'inclusion dans un groupe d'objets dynamique, mais qu'ils n'y soient pas ajoutés. Ce scénario entraînera le succès d'une commande de création ou de mise à jour lors de la journalisation d'une erreur et de la génération d'une `AddThingToDynamicThingGroupsFailed` métrique. Une seule métrique peut représenter plusieurs entrées de journal.

Une entrée du journal des erreurs est créée dans le CloudWatch journal lorsque les événements suivants se produisent :

- Un objet éligible ne peut pas être ajouté à un groupe d'objets dynamique.
- Un objet est supprimé d'un groupe d'objets dynamique pour l'ajouter à un autre groupe.

Lorsqu'un objet peut être ajouté à un groupe d'objets dynamique, tenez compte des points suivants :

- Est-ce que l'objet est déjà contenu dans autant de groupes que possible ? (Consultez la section limites)
  - NON : L'objet est ajouté au groupe d'objets dynamiques.
  - OUI : L'objet est-il un membre d'un groupe d'objets dynamiques ?
    - NON : L'objet ne peut pas être ajouté au groupe d'objets dynamiques, une erreur est enregistrée et une métrique `AddThingToDynamicThingGroupsFailed` est générée.
    - OUI : Le groupe d'objets dynamiques à rejoindre est-il plus ancien que les groupes d'objets dynamiques dont l'objet est déjà membre ?
      - NON : L'objet ne peut pas être ajouté au groupe d'objets dynamiques, une erreur est enregistrée et une métrique `AddThingToDynamicThingGroupsFailed` est générée.
      - OUI : supprimez l'objet du groupe d'objets dynamiques le plus récent, enregistrez une erreur et ajoutez-le au groupe d'objets dynamique. Cela génère une erreur et une métrique `AddThingToDynamicThingGroupsFailed` pour le groupe d'objets dynamiques duquel l'objet a été supprimé.

Lorsqu'un objet d'un groupe d'objets dynamique ne répond plus à la requête de recherche, il est retiré du groupe d'objets dynamique. De même, lorsqu'un objet est mis à jour pour répondre à la requête de recherche d'un groupe d'objets dynamique, l'objet est ensuite ajouté au groupe comme décrit précédemment. Ces ajouts et suppressions sont normaux et ne produisent pas d'entrées de journal des erreurs.

Si l'attribut **`overrideDynamicGroups`** est activé, les groupes statiques sont prioritaires par rapport aux groupes dynamiques

Le nombre de groupes auxquels un objet peut appartenir est limité. Lorsque vous utilisez les UpdateThingGroupsForThingcommandes AddThingToThingGroupou pour mettre à jour l'appartenance à un objet, l'ajout du `--overrideDynamicGroups` paramètre donne la priorité aux groupes d'objets statiques par rapport aux groupes d'objets dynamiques.

Lorsque vous ajoutez un objet à un groupe d'objets statique, tenez compte des points suivants :

- Est-ce que l'objet appartient déjà au nombre maximal de groupes ?
  - NON : La chose est ajoutée au groupe d'objets statiques.
  - OUI : Est-ce que l'objet est contenu dans des groupes dynamiques ?
    - NON : L'objet ne peut pas être ajouté au groupe d'objets. La commande déclenche une exception.

- OUI : Le paramètre --overrideDynamicGroups était-il activé ?
  - NON : L'objet ne peut pas être ajouté au groupe d'objets. La commande déclenche une exception.
  - OUI : L'objet est supprimé du groupe d'objets dynamiques le plus récent, une erreur est enregistrée et une métrique `AddThingToDynamicThingGroupsFailed` est générée pour le groupe d'objets dynamiques dont l'objet a été supprimé. Ensuite, l'objet est ajouté au groupe d'objets statiques.

Les groupes d'objets dynamiques plus anciens sont prioritaires par rapport aux groupes d'objets plus récents.

Le nombre de groupes auxquels un objet peut appartenir est limité. Lorsqu'une opération de création ou de mise à jour crée une éligibilité de groupe supplémentaire pour un objet et que l'objet a atteint sa limite de groupe, il est possible de le supprimer d'un autre groupe d'objets dynamique pour activer cet ajout. Pour plus d'informations sur la façon de procéder, consultez Les commandes réussies peuvent enregistrer les erreurs et Si l'attribut `overrideDynamicGroups` est activé, les groupes statiques sont prioritaires par rapport aux groupes dynamiques pour obtenir des exemples.

Lorsqu'un objet est supprimé d'un groupe d'objets dynamique, une erreur est enregistrée et un événement est déclenché.

Vous ne pouvez pas appliquer de politique à des groupes d'objets dynamiques

Le fait de tenter d'appliquer une stratégie à un groupe d'objets dynamiques génère une exception.

L'appartenance à un groupe d'objets dynamique est cohérente à terme

Seul le dernier état d'un objet est évalué pour le registre. Les états intermédiaires peuvent être ignorés s'ils sont mis à jour rapidement. Évitez d'associer une règle ou une tâche à un groupe d'objets dynamique dont l'appartenance dépend d'un état intermédiaire.

Associer un AWS IoT objet à une connexion client MQTT

Une association d'objets exclusive se produit lorsque vous attachez un certificat X.509 à un seul AWS IoT objet. Dans ce cas, le certificat ne peut pas être utilisé avec d'autres éléments. En garantissant qu'un certificat n'est utilisé que par un seul objet IoT, il permet de prévenir les failles de sécurité.

Dans AWS IoT, l'ID client est un identifiant unique pour un objet ou un appareil lorsqu'il se connecte au courtier AWS IoT Core MQTT. Si vous utilisez une association non exclusive, plusieurs éléments peuvent être associés au même certificat. Lorsqu'une association d'objets non exclusive est en place, afin de maintenir une association claire et d'éviter d'éventuels conflits, vous devez associer votre identifiant client au nom de l'objet.

Dans cette rubrique

- Cas d’utilisation
- Comment associer un objet à une connexion

Cas d’utilisation

L'association d'un objet à une connexion fournit les fonctionnalités suivantes.

> **Note**
>
> Notez que si votre objet IoT et votre connexion client ont une association non exclusive, vous pouvez utiliser toutes les fonctionnalités suivantes, à l'exception de la fonctionnalité des événements du cycle de vie. Pour inclure le nom de votre objet dans les messages relatifs aux événements du cycle de vie, votre objet IoT et votre connexion client doivent avoir une association exclusive.

Variables de politique d'objets : vous pouvez utiliser des variables de politique d'objets pour autoriser l'accès des appareils aux opérations d' AWS IoT API. Ces variables vous permettent de rédiger des AWS IoT Core politiques qui accordent ou refusent des autorisations en fonction des propriétés des objets telles que les noms, les types et les valeurs d'attribut. En utilisant des variables de politique des objets, vous pouvez appliquer la même stratégie pour contrôler plusieurs AWS IoT Core appareils. Cela vous permet de simplifier la gestion des politiques et de réduire la duplication des ressources. Pour plus d'informations, consultez la section Variables de politique des objets.

Événements du cycle de vie : vous pouvez recevoir le nom de l'objet dans les événements du cycle de vie (par exemple, connexion, déconnexion, abonnement et désinscription). Cela permet de traiter le nom de l'objet inclus dans les messages, par exemple dans les règles. Pour plus d'informations, consultez la section Événements du cycle de vie.

Journalisation spécifique aux ressources : vous pouvez configurer la journalisation spécifique aux ressources pour les groupes d'objets et appliquer facilement la configuration de journalisation

souhaitée pour tous les éléments du groupe d'objets défini. Pour de plus amples informations, veuillez consulter ???.

Répartition des coûts : vous pouvez créer des groupes de facturation avec des balises personnalisées pour la répartition des coûts et ajouter les éléments à ces groupes. Pour plus d'informations, consultez la section Groupes de facturation.

Comment associer un objet à une connexion

Si votre ID client correspond au nom de votre objet dans le registre, une fois que vous aurez attaché un certificat X.509 à cet objet IoT, la connexion client AWS IoT Core sera associée à l'objet. Si votre ID client ne correspond pas au nom de l'objet dans le registre, vous pouvez exclusivement associer un certificat X.509 à l'objet pour établir cette association. La chose qui possède cet attachement exclusif est appelée chose exclusive. Sinon, c'est ce qu'on appelle une chose non exclusive. Lorsqu'un certificat est associé à un objet exclusif, il ne peut être associé à d'autres éléments que si vous le détachez de l'objet exclusif. Dans cette section, choisissez AWS Management Console soit AWS CLI d'associer un objet à une connexion.

AWS Management Console

Pour joindre un certificat à un objet en utilisant exclusivement le AWS Management Console.

1. Ouvrez la page d'AWS IoT accueil dans la AWS IoT console. Dans le menu de navigation de gauche, dans Sécurité, sélectionnez Certificats.
2. Sur la page Certificats, choisissez le certificat auquel vous souhaitez associer un objet. Choisissez ensuite Attacher à des objets dans Actions dans le coin supérieur droit de la page.

   Vous pouvez également choisir un certificat et accéder à la page de détails du certificat. Choisissez l'onglet Objets, puis choisissez Joindre à des objets.
3. Sur la page Joindre un certificat à un ou plusieurs objets, cochez la case Associer un objet à une connexion. Choisissez ensuite un objet auquel joindre ce certificat dans la liste déroulante des objets.
4. Choisissez Joindre un ou plusieurs objets. Si l'action aboutit, vous verrez une bannière indiquant « Vous avez correctement attaché un objet à votre certificat », et l'objet sera ajouté à l'onglet Objets.

Pour détacher un certificat d'un objet exclusif à l'aide du AWS Management Console

1. Ouvrez la page d'AWS IoT accueil dans la AWS IoT console. Dans le menu de navigation de gauche, dans Sécurité, sélectionnez Certificats.
2. Sur la page Certificats, choisissez un certificat et accédez à la page de détails du certificat.
3. Sur la page des détails du certificat, choisissez l'onglet Objets. Choisissez ensuite l'élément auquel vous souhaitez détacher le certificat. Choisissez Détacher les objets.
4. Dans la fenêtre Détacher les objets, confirmez votre action. Choisissez Détacher. Si l'action aboutit, vous verrez une bannière indiquant « Dissociation réussie d'un objet de votre certificat », et l'objet n'apparaîtra plus dans l'onglet Objets.

AWS CLI

1. Pour associer un certificat à un objet en utilisant AWS CLI, exécutez la attach-thing-principalcommande. Pour spécifier la certificate-to-thing pièce jointe exclusive, vous devez le spécifier `EXCLUSIVE_THING` dans le `--thing-principal-type` champ. Voici un exemple de commande.

   ```
   aws iot attach-thing-principal \
       --thing-name "thing_1" \
       --principal "arn:aws:iot:us-
   east-1:123456789012:cert/
   2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
       --thing-principal-type "EXCLUSIVE_THING"
   ```

   Cette commande ne produit aucune sortie. Pour de plus amples informations, veuillez consulter ???.
2. Pour répertorier les éléments associés au certificat spécifié ainsi que le type de pièce jointe, exécutez la `list-principal-things-v2` commande. Le type de pièce jointe fait référence à la manière dont le certificat est attaché à l'objet. Voici un exemple de commande.

   ```
   $ aws iot list-principal-things-v2 \
       --principal "arn:aws:iot:us-
   east-1:123456789012:cert/
   2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
   ```

   Le résultat peut ressembler à ce qui suit.

```
{
    "PrincipalThingObjects": [
        {
            "thingPrincipalType": "EXCLUSIVE_THING",
            "thing": "arn:aws:iot:us-east-1:123456789012:thing/thing_1"
        }
    ]
}
```

Pour de plus amples informations, veuillez consulter ???.

3. Pour répertorier les principaux associés à l'objet spécifié ainsi que le type de pièce jointe, exécutez la `list-thing-principals-v2` commande. Le type de pièce jointe fait référence à la manière dont le certificat est attaché à l'objet. Voici un exemple de commande.

```
$ aws iot list-thing-principals-v2 \
    --thing-name "thing_1"
```

Le résultat peut ressembler à ce qui suit.

```
{
    "ThingPrincipalObjects": [
        {
            "thingPrincipalType": "EXCLUSIVE_THING",
            "principal": "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8"
        }
    ]
}
```

Pour de plus amples informations, veuillez consulter ???.

4. Pour détacher un certificat d'un objet, exécutez la detach-thing-principalcommande.

```
aws iot detach-thing-principal \
    --principal "arn:aws:iot:us-
east-1:123456789012:cert/
2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8" \
    --thing-name "thing_1"
```

Cette commande ne produit aucune sortie. Pour de plus amples informations, veuillez consulter ???.

Ajout d'attributs de propagation pour l'enrichissement des messages

Dans AWS IoT Core, vous pouvez enrichir les messages MQTT des appareils en ajoutant des attributs de propagation, qui sont des métadonnées contextuelles issues d'attributs d'objets ou de détails de connexion. Ce processus, connu sous le nom d'enrichissement des messages, peut être utile dans divers scénarios. Par exemple, vous pouvez enrichir les messages pour chaque opération de publication entrante sans apporter de modifications côté appareil ni utiliser de règles. En tirant parti des attributs de propagation, vous pouvez bénéficier d'un moyen plus efficace et plus rentable d'enrichir vos données IoT sans les complexités liées à la configuration des règles ou à la gestion des configurations de republication.

La fonctionnalité d'enrichissement des messages est disponible pour AWS IoT Core les clients qui utilisent Basic Ingest et Message Broker. Il est important de noter que même si les appareils de publication peuvent utiliser n'importe quelle version de MQTT, les abonnés (applications ou services consommant des messages) doivent prendre en charge MQTT 5 pour recevoir les messages enrichis avec des attributs de propagation. Les messages enrichis seront ajoutés en tant que propriétés utilisateur MQTT 5 à chaque message publié depuis des appareils. Si vous utilisez des règles, vous pouvez tirer parti de la fonction get_user_properties pour récupérer les données enrichies pour le routage ou le traitement des messages en fonction de ces données.

Dans AWS IoT Core, vous pouvez ajouter des attributs de propagation lorsque vous créez ou mettez à jour un type d'objet, en utilisant le AWS Management Console ou le AWS CLI.

> **Important**
>
> Lorsque vous ajoutez des attributs de propagation, vous devez vous assurer que le client qui publie le message a été authentifié par un certificat. Pour de plus amples informations, veuillez consulter Authentification client.

> **Note**
>
> Si vous essayez de tester cette fonctionnalité à l'aide du client de test MQTT dans la console, elle risque de ne pas fonctionner car cette fonctionnalité nécessite que les clients MQTT soient authentifiés avec un certificat associé.

AWS Management Console

Pour ajouter des attributs de propagation destinés à enrichir les messages à l'aide du AWS Management Console

1. Ouvrez la page d'AWS IoT accueil dans la AWS IoT console. Dans le volet de navigation de gauche, dans Gérer, sélectionnez Tous les appareils. Choisissez ensuite les types d'objets.
2. Sur la page Types d'objets, choisissez Créer un type d'objet.

   Pour configurer l'enrichissement des messages en mettant à jour un type d'objet, choisissez-en un. Ensuite, sur la page de détails du type d'objet, choisissez Mettre à jour.
3. Sur la page Créer un type d'objet, choisissez ou entrez les informations relatives au type d'objet dans les propriétés du type d'objet.

   Si vous choisissez de mettre à jour un type d'objet, les propriétés du type d'objet apparaîtront une fois que vous aurez sélectionné Mettre à jour à l'étape précédente.
4. Dans Configuration supplémentaire, développez les attributs de propagation. Choisissez ensuite Attribut d'objet et entrez l'attribut d'objet que vous souhaitez renseigner dans les MQTT5 messages publiés. À l'aide de la console, vous pouvez ajouter jusqu'à trois attributs d'objets.

   Dans la section Attributs de propagation, choisissez Attribut de connexion et entrez le type d'attribut et éventuellement le nom de l'attribut.
5. Ajoutez éventuellement des balises. Choisissez ensuite Créer un type d'objet.

   Si vous choisissez de mettre à jour un type d'objet, choisissez Mettre à jour le type d'objet.

AWS CLI

1. Pour ajouter des attributs de propagation destinés à enrichir les messages en créant un nouveau type d'objet à l'aide de AWS CLI, exécutez la create-thing-typecommande. Voici un exemple de commande.

```
aws iot create-thing-type \
    --thing-type-name "LightBulb" \
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

   Le résultat de la commande peut ressembler à ce qui suit.

```
{
 "thingTypeName": "LightBulb",
 "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
 "thingTypeId": "ce3573b0-0a3c-45a7-ac93-4e0ce14cd190"
}
```

2. Pour configurer l'enrichissement des messages en mettant à jour un type d'objet à l'aide de AWS CLI, exécutez la update-thing-typecommande. Notez que vous ne pouvez effectuer la mise à jour que `mqtt5Configuration` lorsque vous exécutez cette commande. Voici un exemple de commande.

```
aws iot update-thing-type \
    --thing-type-name "MyThingType" \
    --thing-type-properties "{\"mqtt5Configuration\":{\"propagatingAttributes\":
[{\"userPropertyKey\":\"iot:ClientId\", \"connectionAttribute\":\"iot:ClientId\"},
 {\"userPropertyKey\":\"test\", \"thingAttribute\":\"A\"}]}}" \
```

   Cette commande ne produit aucune sortie.

3. Pour décrire un type d'objet, exécutez la `describe-thing-type` commande. Cette commande produira une sortie contenant des informations de configuration d'enrichissement des messages sur le `thing-type-properties` terrain. Voici un exemple de commande.

```
aws iot describe-thing-type \
    --thing-type-name "LightBulb"
```

Le résultat peut ressembler à ce qui suit.


```
{
 "thingTypeName": "LightBulb",
 "thingTypeId": "bdf72512-0116-4392-8d79-bf39b17ef73d",
 "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/LightBulb",
 "thingTypeProperties": {
  "mqtt5Configuration": {
   "propagatingAttributes": [
    {
     "userPropertyKey": "iot:ClientId",
     "connectionAttribute": "iot:ClientId"
    },
    {
     "userPropertyKey": "test",
     "thingAttribute": "attribute"
    }
   ]
  }
 },
 "thingTypeMetadata": {
  "deprecated": false,
  "creationDate": "2024-10-18T17:37:46.656000+00:00"
 }
}
```


Pour de plus amples informations, veuillez consulter ???.

Marquer vos ressources AWS IoT

Pour vous aider à gérer et à organiser vos groupes d'objets, vos types d'objet, vos règles de rubrique, vos tâches, vos audits planifiés ainsi que vos profils de sécurité, vous pouvez, le cas échéant, attribuer vos propres métadonnées à chacune de ces ressources sous la forme de balises. Cette section décrit les balises et vous montre comment les créer.

Pour vous aider à gérer vos coûts liés aux objets, vous pouvez créer des groupes de facturation qui contiennent des objets. Vous pouvez ensuite affecter des balises contenant vos métadonnées à chacun de ces groupes de facturation. Cette section décrit également les groupes de facturation, ainsi que les commandes disponibles pour les créer et les gérer.

Principes de base des étiquettes

Vous pouvez utiliser des balises pour classer vos AWS IoT ressources de différentes manières (par exemple, par objectif, propriétaire ou environnement). Cette approche est utile lorsque vous avez de nombreuses ressources du même type : elle vous permet d'identifier rapidement une ressource en fonction des balises que vous lui avez attribuées. Chaque balise est constituée d'une clé et d'une valeur facultative que vous définissez. Par exemple, vous pouvez définir un ensemble de balises pour vos types d'objet vous permettant de suivre les appareils par type. Nous vous recommandons de créer un ensemble de clés de balise répondant à vos besoins pour chaque type de ressource. L'utilisation d'un ensemble de clés de balise cohérent facilite la gestion de vos ressources.

Vous pouvez rechercher et filtrer les ressources en fonction des balises que vous ajoutez ou appliquez. Vous pouvez également utiliser des balises de groupe de facturation pour classer les coûts par catégorie et en effectuer le suivi. Vous pouvez également utiliser des balises pour contrôler l'accès à vos ressources, comme décrit dans Utilisation des balises avec des politiques IAM.

Pour faciliter l'utilisation, l'éditeur de balises de la console AWS de gestion fournit un moyen centralisé et unifié de créer et de gérer vos balises. Pour plus d'informations, consultez la section Utilisation de l'éditeur de balises dans Utilisation de la console AWS de gestion.

Vous pouvez également travailler avec des balises à l'aide de l'API AWS CLI et de l' AWS IoT API. Vous pouvez associer des balises à des groupes d'objets, des types d'objet, des règles de rubrique, des tâches, des profils de sécurité ainsi que des groupes de facturation lorsque vous les créez en utilisant `Tags` le champ dans les commandes suivantes :

- CreateBillingGroup

- CreateDestination
- CreateDeviceProfile
- CreateDynamicThingGroup
- CreateJob
- CréerOTAUpdate
- CreatePolicy
- CreateScheduledAudit
- CreateSecurityProfile
- CreateServiceProfile
- CreateStream
- CreateThingGroup
- CreateThingType
- CreateTopicRule
- CreateWirelessGateway
- CreateWirelessDevice

Vous pouvez ajouter, modifier ou supprimer des balises pour les ressources existantes qui prennent en charge le balisage à l'aide des commandes suivantes :

- TagResource
- ListTagsForResource
- UntagResource

Vous pouvez modifier les clés et valeurs de balise, et vous pouvez retirer des balises d'une ressource à tout moment. Vous pouvez définir la valeur d'une balise sur une chaîne vide, mais vous ne pouvez pas définir la valeur d'une balise sur null. Si vous ajoutez une balise ayant la même clé qu'une balise existante sur cette ressource, la nouvelle valeur remplace l'ancienne valeur. Si vous supprimez une ressource, toutes les balises associées à celle-ci sont également supprimées.

Limites et restrictions liées aux balises

Les restrictions de base suivantes s'appliquent aux balises :

- Nombre maximal de balises par ressource : 50

- Longueur de clé maximale : 127 caractères Unicode en UTF-8
- Longueur de valeur maximale : 255 caractères Unicode en UTF-8
- Les clés et valeurs de balise sont sensibles à la casse.
- N'utilisez pas le `aws:` préfixe dans vos noms et valeurs de balise. Il est réservé à AWS l'usage. Vous ne pouvez pas modifier ou supprimer des noms ou valeurs de balise ayant ce préfixe. Les balises avec ce préfixe ne sont pas comptabilisées comme vos balises pour la limite de ressources.
- Si votre schéma de balisage est utilisé pour plusieurs services et ressources, n'oubliez pas que d'autres services peuvent avoir des restrictions concernant les caractères autorisés. Les caractères autorisés incluent les lettres, les espaces et les chiffres représentables en UTF-8, ainsi que les caractères spéciaux suivants : + - = . _ : / @.

Utilisation des balises avec des politiques IAM

Vous pouvez appliquer des autorisations de niveau ressource basées sur des balises dans les stratégies IAM que vous utilisez pour les actions d'API AWS IoT . Vous bénéficiez ainsi d'un meilleur contrôle sur les ressources qu'un utilisateur peut créer, modifier ou utiliser. Vous pouvez utiliser l'élément `Condition` (également appelé bloc `Condition`) avec les clés et valeurs de contexte de condition suivantes dans une politique IAM pour contrôler l'accès des utilisateurs (autorisations) en fonction des balises d'une ressource :

- Utilisez `aws:ResourceTag/`*`tag-key`*`: `*`tag-value`* pour accorder ou refuser aux utilisateurs des actions sur des ressources ayant des balises spécifiques.
- Utilisez `aws:RequestTag/`*`tag-key`*`: `*`tag-value`* pour exiger qu'une balise spécifique soit utilisée (ou ne soit pas utilisée) lorsque vous effectuez une demande d'API pour créer ou modifier une ressource qui autorise les balises.
- Utilisez `aws:TagKeys: [`*`tag-key`*`, ...]` pour exiger qu'un ensemble de clés de balise spécifique soit utilisé (ou ne soit pas utilisé) lorsque vous effectuez une demande d'API pour créer ou modifier une ressource qui autorise les balises.

> **Note**
>
> Les clés et valeurs de contexte de condition d'une politique IAM s'appliquent uniquement aux AWS IoT actions pour lesquelles un identifiant pour une ressource susceptible d'être étiquetée est un paramètre obligatoire. Par exemple, l'utilisation de n'DescribeEndpointest pas autorisée ou refusée sur la base des clés et des valeurs du contexte des conditions, car

aucune ressource balisable (groupes d'objets, types d'objets, règles de rubrique, tâches ou profil de sécurité) n'est référencée dans cette demande. Pour plus d'informations sur les AWS IoT ressources balisables et les clés de condition qu'elles prennent en charge, consultez la section Actions, ressources et clés de condition pour. AWS IoT

Pour de plus amples informations, veuillez consulter Contrôle de l'accès à l'aide d'étiquettes dans le AWS Identity and Access Management Guide de l'utilisateur. La section Référence de stratégie JSON IAM de ce guide fournit la syntaxe détaillée, des descriptions, ainsi que des exemples des éléments, des variables et de la logique d'évaluation des stratégies JSON dans IAM.

L'exemple de stratégie suivant applique deux restrictions basées sur des `ThingGroup` actions. Un utilisateur IAM restreint par cette stratégie :

- Impossible de créer un groupe d'objets avec la balise « env=prod » (dans l'exemple, voir la ligne`"aws:RequestTag/env" : "prod"`).
- Impossible de modifier ou d'accéder à un groupe de choses qui a une balise existante "env=prod" (dans l'exemple, voir la ligne `"aws:ResourceTag/env" : "prod"`).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "iot:CreateThingGroup",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
```

```
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

Vous pouvez également spécifier plusieurs valeurs de balise pour une clé de balise donnée en les plaçant dans une liste, comme suit :

```
        "StringEquals" : {
          "aws:ResourceTag/env" : ["dev", "test"]
        }
```

> **Note**
>
> Si vous autorisez ou refusez à des utilisateurs l'accès à des ressources en fonction de balises, vous devez envisager de refuser de manière explicite la possibilité pour les utilisateurs d'ajouter ces balises ou de les supprimer des mêmes ressources. Sinon, il sera possible pour un utilisateur de contourner vos restrictions et d'obtenir l'accès à une ressource en modifiant ses balises.

Groupes de facturation

AWS IoT ne vous permet pas d'appliquer directement des balises à des éléments individuels, mais vous permet de placer des éléments dans des groupes de facturation et de leur appliquer des balises. En effet AWS IoT, l'allocation des données de coût et d'utilisation en fonction des balises est limitée aux groupes de facturation.

AWS IoT Core pour les ressources LoRa WAN, telles que les appareils sans fil et les passerelles, ne peuvent pas être ajoutées aux groupes de facturation. Ils peuvent toutefois être associés à des AWS IoT éléments qui peuvent être ajoutés aux groupes de facturation.

Les commandes suivantes sont disponibles :

- AddThingToBillingGroup ajoute un objet à un groupe de facturation.
- CreateBillingGroup crée un groupe de facturation.
- DeleteBillingGroup supprime le groupe de facturation.
- DescribeBillingGroup renvoie des informations sur un groupe de facturation.
- ListBillingGroups répertorie les groupes de facturation que vous avez créés.
- ListThingsInBillingGroup répertorie les objets que vous avez ajoutés au groupe de facturation donné.
- RemoveThingFromBillingGroup supprime l'objet indiqué du groupe de facturation.
- UpdateBillingGroup met à jour les informations sur le groupe de facturation.
- CreateThing vous permet de spécifier un groupe de facturation pour l'objet lorsque vous le créez.
- DescribeThing renvoie la description d'un objet, y compris le groupe de facturation auquel appartient l'objet, le cas échéant.

L'API AWS IoT Wireless fournit ces actions pour associer des appareils et des passerelles sans fil à des AWS IoT objets.

- AssociateWirelessDeviceWithThing
- AssociateWirelessGatewayWithThing

Affichage des données de répartition des coûts et d'utilisation

Vous pouvez utiliser des balises de groupe de facturation pour classer les coûts par catégorie et en effectuer le suivi. Lorsque vous appliquez des balises aux groupes de facturation (et donc aux

éléments qu'ils incluent), AWS génère un rapport de répartition des coûts sous forme de fichier CSV (valeurs séparées par des virgules) avec votre utilisation et vos coûts agrégés par vos balises. Vous pouvez appliquer des balises associées à des catégories métier (telles que les centres de coûts, les noms d'applications ou les propriétaires) pour organiser les coûts relatifs à divers services. Pour de plus amples informations sur l'utilisation des balises pour la répartition des coûts, veuillez consulter Utilisation des balises de répartition des coûts dans le Guide de l'utilisateur AWS Billing and Cost Management.

> **Note**
>
> Pour associer avec précision les données d'utilisation et de coût aux objets que vous avez placés dans des groupes de facturation, chaque appareil ou application doit :
>
> - Être enregistré en tant qu'objet dans AWS IoT. Pour de plus amples informations, veuillez consulter Gestion des appareils avec AWS IoT.
> - Connectez-vous au courtier de AWS IoT messages via MQTT en utilisant uniquement le nom de l'objet comme ID client. Pour de plus amples informations, veuillez consulter the section called “Protocoles de communication des appareils”. Si votre identifiant client ne correspond pas au nom de l'objet, vous pouvez activer la pièce jointe exclusive pour établir l'association. Pour de plus amples informations, veuillez consulter ???.
> - S'authentifier à l'aide d'un certificat client associé à l'objet.

Les dimensions de tarification suivantes sont disponibles pour les groupes de facturation (en fonction de l'activité des objets associés au groupe de facturation) :

- Connectivité (en fonction du nom d'objet utilisé comme ID client pour la connexion).
- Messagerie (en fonction des messages entrants provenant d'un objet et sortants vers un objet, MQTT uniquement).
- Opérations de shadow (en fonction de l'objet dont le message a déclenché une mise à jour de shadow).
- Règles déclenchées (selon l'objet dont le message entrant a déclenché la règle ; ne s'applique pas aux règles déclenchées par des événements de cycle de vie MQTT).
- Mises à jour d'index d'objets (en fonction de l'objet qui a été ajouté à l'index).
- Actions distantes (en fonction de l'objet mis à jour).
- AWS IoT Device Defender détecter les rapports (en fonction de l'objet dont l'activité est signalée).

Les données d'utilisation et de coût basées sur des balises (et signalées pour un groupe de facturation) ne reflètent pas les activités suivantes :

- Opérations de registre d'appareils (y compris les mises à jour d'objets, de groupes d'objets et de types d'objet). Pour de plus amples informations, veuillez consulter Gestion des appareils avec AWS IoT).
- Mises à jour d'index de groupes d'objets (lors de l'ajout d'un groupe d'objets).
- Requêtes de recherche d'index.
- Mise en service des appareils.
- AWS IoT Device Defender rapports d'audit.

Sécurité dans AWS IoT

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS IoT, consultez la section AWS Services concernés par programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre entreprise, ainsi que la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de son utilisation AWS IoT. Les rubriques suivantes expliquent comment procéder à la configuration AWS IoT pour atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos AWS IoT ressources.

Rubriques

- AWS IoT sécurité
- Authentification
- Autorisation
- Protection des données dans AWS IoT Core
- Gestion des identités et des accès pour AWS IoT
- Journalisation et surveillance
- Validation de conformité pour AWS IoT Core
- La résilience au cœur de AWS l'IoT
- Utilisation AWS IoT Core avec des points de terminaison VPC d'interface

- Sécurité de l'infrastructure dans AWS IoT
- Surveillance de la sécurité des flottes de production ou des appareils avec Core AWS IoT
- Bonnes pratiques en matière de sécurité dans AWS IoT Core
- AWS formation et certification

AWS IoT sécurité

Chaque appareil ou client connecté doit disposer d'informations d'identification pour interagir avec AWS IoT. Tout le trafic en provenance et à destination AWS IoT est envoyé de manière sécurisée via le protocole TLS (Transport Layer Security). AWS les mécanismes de sécurité du cloud protègent les données lorsqu'elles passent AWS IoT d'un AWS service à un autre.

- Vous êtes responsable de la gestion des informations d'identification des appareils (certificats X.509, informations d'identification, Amazon Cognito, identités AWS , identités fédérées ou jetons d'authentification personnalisés) et des politiques dans AWS IoT. Pour de plus amples informations, veuillez consulter Gestion des clés dans AWS IoT. Vous êtes responsable de l'affectation d'identités uniques à chaque appareil et de la gestion des autorisations de chaque appareil ou groupe d'appareils.
- Vos appareils se connectent à AWS IoT l'aide de certificats X.509 ou d'identités Amazon Cognito via une connexion TLS sécurisée. Au cours de la recherche et du développement, et pour certaines applications qui appellent ou utilisent des API WebSockets, vous pouvez également vous authentifier à l'aide d'utilisateurs et de groupes IAM ou de jetons d'authentification personnalisés. Pour de plus amples informations, veuillez consulter Utilisateurs, groupes et rôles IAM.

- Lorsque vous utilisez l' AWS IoT authentification, le courtier de messages est chargé d'authentifier vos appareils, d'ingérer en toute sécurité les données des appareils et d'accorder ou de refuser les autorisations d'accès que vous spécifiez pour vos appareils à l'aide AWS IoT de politiques.
- Lorsque vous utilisez l'authentification personnalisée, un autorisateur personnalisé est chargé d'authentifier vos appareils et d'accorder ou de refuser les autorisations d'accès que vous spécifiez pour vos appareils en utilisant les politiques AWS IoT IAM.
- Le moteur de AWS IoT règles transmet les données des appareils à d'autres appareils ou à d'autres AWS services conformément aux règles que vous définissez. Il permet AWS Identity and Access Management de transférer des données en toute sécurité vers leur destination finale. Pour de plus amples informations, veuillez consulter Gestion des identités et des accès pour AWS IoT.

Authentification

L'authentification est un mécanisme permettant de vérifier l'identité d'un client ou d'un serveur. L'authentification du serveur est le processus par lequel les appareils ou autres clients s'assurent qu'ils communiquent avec un point de AWS IoT terminaison réel. L'authentification des clients est le processus par lequel les appareils ou d'autres clients s'authentifient. AWS IoT

Présentation des certificats X.509

Les certificats X.509 sont des certificats numériques qui font appel à la norme d'infrastructure de clé publique X.509 pour associer une clé publique à une identité contenue dans un certificat. Les certificats X.509 sont émis par une entité de confiance appelée autorité de certification (CA). Celle-ci gère un ou plusieurs certificats spéciaux appelés certificats d'autorité de certification, qu'elle utilise pour émettre des certificats X.509. Seule l'autorité de certification a accès aux certificats d'autorité de certification (CA). Les chaînes de certificats X.509 sont utilisées à la fois pour l'authentification du serveur par les clients et pour l'authentification du client par le serveur.

Authentification du serveur

Lorsque votre appareil ou un autre client tente de se connecter AWS IoT Core, le AWS IoT Core serveur envoie un certificat X.509 que votre appareil utilise pour authentifier le serveur. L'authentification a lieu au niveau de la couche TLS par le biais de la validation de la chaîne du certificat X.509. Il s'agit de la même méthode que celle utilisée par votre navigateur lorsque vous visitez une URL HTTPS. Si vous souhaitez utiliser des certificats de votre propre autorité de certification, veuillez consulter Gestion de vos certificats d’autorité de certification.

Lorsque vos appareils ou d'autres clients établissent une connexion TLS avec un point de AWS IoT Core terminaison, AWS IoT Core présente une chaîne de certificats que les appareils utilisent pour vérifier qu'ils communiquent avec eux AWS IoT Core et qu'aucun autre serveur ne se fait passer AWS IoT Core pour un autre serveur. La chaîne présentée dépend de la combinaison du type de point de terminaison auquel l'appareil se connecte et de la suite de chiffrement AWS IoT Core négociée par le client lors de la prise de contact TLS.

Types de point de terminaison

AWS IoT Core soutient`iot:Data-ATS`. `iot:Data-ATS`les points de terminaison présentent un certificat de serveur signé par une autorité de certification Amazon Trust Services.

Les certificats présentés par les points de terminaison ATS ont été signés par Starfield (signature croisée). Certaines mises en œuvre de client TLS nécessitent la validation de la racine de confiance et exigent que les certificats d'autorité de certification Starfield soient installés dans les magasins d'approbations du client.

> **Warning**
>
> L'utilisation d'une méthode d'épinglage de certificat qui hache l'ensemble du certificat (y compris le nom de l'émetteur, etc.) n'est pas recommandée car cela entraînera l'échec de la vérification du certificat, étant donné que les certificats ATS que nous fournissons sont signés par Starfield (signature croisée) et ont un nom d'émetteur différent.

> **Important**
>
> Utilisez des `iot:Data-ATS` points de terminaison. Les certificats Symantec et Verisign sont devenus obsolètes et ne sont plus pris en charge par. AWS IoT Core

Vous pouvez utiliser la commande `describe-endpoint` pour créer votre point de terminaison ATS.

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

Cette commande `describe-endpoint` renvoie un point de terminaison au format suivant.

```
account-specific-prefix.iot.your-region.amazonaws.com
```

> **Note**
>
> La première fois que la commande `describe-endpoint` est appelée, un point de terminaison est créé. Tous les appels suivants de `describe-endpoint` renvoient le même point de terminaison.

> **Note**
>
> Pour afficher votre `iot:Data-ATS` point de terminaison dans la AWS IoT Core console, choisissez Paramètres. La console affiche uniquement le point de terminaison `iot:Data-ATS`.

Création d'un **`IotDataPlaneClient`** avec le AWS SDK pour Java

Pour créer une annonce `IotDataPlaneClient` utilisant un `iot:Data-ATS` point de terminaison, vous devez effectuer les opérations suivantes.

- Créez un `iot:Data-ATS` point de terminaison à l'aide de l'DescribeEndpointAPI.
- Spécifiez ce point de terminaison lorsque vous créez le `IotDataPlaneClient`.

L'exemple suivant exécute ces deux opérations.

```
public void setup() throws Exception {
        IotClient client =
 IotClient.builder().credentialsProvider(CREDENTIALS_PROVIDER_CHAIN).region(Region.US_EAST_1).b
        String endpoint = client.describeEndpoint(r -> r.endpointType("iot:Data-
ATS")).endpointAddress();
        iot = IotDataPlaneClient.builder()
                                .credentialsProvider(CREDENTIALS_PROVIDER_CHAIN)
                                .endpointOverride(URI.create("https://" + endpoint))
                                .region(Region.US_EAST_1)
                                .build();
}
```

Certificats d'autorité de certification pour l'authentification du serveur

Selon le type de point de terminaison de données que vous utilisez et la suite de chiffrement que vous avez négociée, les certificats d'authentification AWS IoT Core du serveur sont signés par l'un des certificats d'autorité de certification racine suivants :

Points de terminaison Amazon Trust Services (préférés)

> **Note**
>
> Vous devrez peut-être faire un clic droit sur ces liens et sélectionner Enregistrer le lien sous... pour enregistrer ces certificats sous forme de fichiers.

- Clé RSA 2048 bits : Amazon Root CA 1
- Clé RSA 4096 bits : Amazon Root CA 2 Réservé pour un usage futur.
- Clé ECC 256 bits : Amazon Root CA 3
- Clé ECC 384 bits : Amazon Root CA 4 Réservé pour un usage futur.

Ces certificats sont tous signés (signature croisée) par le certificat d'autorité de certification racine Starfield. Toutes les nouvelles AWS IoT Core régions, à compter du lancement de AWS IoT Core la région Asie-Pacifique (Mumbai) le 9 mai 2018, ne proposent que des certificats ATS.

VeriSign Points de terminaison (anciens)

- Clé RSA 2048 bits : certificat CA VeriSign racine G5 public principal de classe 3

Instructions d'authentification du serveur

De nombreuses variables peuvent affecter la capacité d'un appareil à valider le certificat d'authentification du serveur AWS IoT Core . Par exemple, les appareils peuvent être trop limités en mémoire pour contenir tous les certificats d'autorité de certification racine possibles, ou les appareils peuvent mettre en œuvre une méthode non standard de validation de certificat. Pour ces raisons, nous suggérons de suivre les instructions suivantes :

- Nous vous recommandons d'utiliser votre point de terminaison ATS et d'installer tous les certificats Amazon Root CA pris en charge.

- Si vous ne pouvez pas stocker tous ces certificats sur votre appareil et si vos appareils n'utilisent pas la validation basée sur ECC, vous pouvez omettre les certificats Amazon Root CA 3 et Amazon Root CA 4 ECC. Si vos appareils ne mettent pas en œuvre la validation de certificats basée sur RSA, vous pouvez omettre les certificats RSA Amazon Root CA 1 et Amazon Root CA 2. Vous devrez peut-être faire un clic droit sur ces liens et sélectionner Enregistrer le lien sous... pour enregistrer ces certificats sous forme de fichiers.
- Si vous rencontrez des problèmes de validation de certificat de serveur lorsque vous vous connectez à votre point de terminaison ATS, essayez d'ajouter le certificat d'autorité de certification racine Amazon à signature croisée pertinent à votre magasin d'approbations. Vous devrez peut-être faire un clic droit sur ces liens et sélectionner Enregistrer le lien sous... pour enregistrer ces certificats sous forme de fichiers.
  - Dument Signé Amazon Root CA 1
  - Dument Signé Amazon Root CA 2 - Réservé pour une utilisation ultérieure.
  - Dument Signé Amazon Root CA 3
  - Dument Signé - Réservé pour une utilisation ultérieure.
- Si vous rencontrez des problèmes de validation de certificat de serveur, votre appareil devra peut-être explicitement approuver l'autorité de certification racine. Essayez d'ajouter Starfield Root CA Certificate à votre boutique de confiance.
- Si vous rencontrez toujours des problèmes après avoir exécuté les étapes ci-dessus, contactez AWS l’assistance aux développeurs.

> **Note**
>
> Les certificats CA ne peuvent pas être utilisés au-delà de leur date d'expiration pour valider un certificat de serveur. Les certificats CA peuvent devoir être remplacés avant leur date d'expiration. Vérifiez que vous pouvez mettre à jour les certificats d’autorité de certification racine sur tous vos appareils ou vos clients afin d'assurer une connectivité permanente et la conformité aux bonnes pratiques de sécurité du moment.

> **Note**
>
> Lorsque vous vous connectez AWS IoT Core au code de votre appareil, transmettez le certificat à l'API que vous utilisez pour vous connecter. L'API que vous utilisez varie selon le kit SDK. Pour plus d'informations, consultez la section AWS IoT Core Appareil SDKs.

Authentification client

AWS IoT prend en charge trois types de principes d'identité pour l'authentification des appareils ou des clients :

- Certificats client X.509
- Utilisateurs, groupes et rôles IAM
- Identités Amazon Cognito

Ces identités peuvent être utilisées avec des appareils et des applications mobiles, Web ou de bureau. Ils peuvent même être utilisés par un utilisateur qui saisit des commandes d'interface de ligne de AWS IoT commande (CLI). Généralement, AWS IoT les appareils utilisent des certificats X.509, tandis que les applications mobiles utilisent les identités Amazon Cognito. Les applications Web et de bureau utilisent IAM ou des identités fédérées. Les commandes AWS CLI utilisent IAM. Pour plus d'informations sur les identités IAM, consultez Gestion des identités et des accès pour AWS IoT.

Certificats client X.509

Les certificats X.509 permettent AWS IoT d'authentifier les connexions entre les clients et les appareils. Les certificats clients doivent être enregistrés pour AWS IoT qu'un client puisse communiquer avec lui AWS IoT. Un certificat client peut être enregistré en plusieurs Compte AWS exemplaires Région AWS pour faciliter le déplacement d'appareils entre vos appareils Compte AWS situés dans la même région. Pour plus d’informations, consultez Utilisation de certificats client X.509 dans plusieurs Compte AWS s avec enregistrement multi-comptes.

Nous vous recommandons d’attribuer à chaque appareil ou client un certificat unique de manière à permettre des actions de gestion des clients bien définies, y compris la révocation de certificats. Les appareils et les clients doivent également prendre en charge la rotation et le remplacement des certificats afin d'aider à assurer un bon fonctionnement à l'expiration de ces derniers.

Pour de plus amples informations sur l'utilisation des certificats X.509 pour la prise en charge d'un grand nombre d'appareils, veuillez consulter Mise en service des appareils pour prendre connaissance des différentes options de gestion des certificats et de mise en service prises en charge par AWS IoT .

AWS IoT prend en charge les types de certificats clients X.509 suivants :

- Certificats X.509 générés par AWS IoT
- Certificats X.509 signés par une autorité de certification enregistrée auprès AWS IoT de.
- Certificats X.509 signés par une autorité de certification non enregistrée auprès d' AWS IoT.

Cette section décrit comment gérer les certificats X.509 dans AWS IoT. Vous pouvez utiliser la AWS IoT console ou AWS CLI effectuer les opérations de certificat suivantes :

- Création de certificats AWS IoT clients
- Création de vos propres certificats clients
- Enregistrement d'un certificat client
- Activation ou désactivation d'un certificat client
- Révocation d'un certificat client

Pour plus d'informations sur les AWS CLI commandes qui exécutent ces opérations, consultez la section Référence de la AWS IoT CLI.

Utilisation des certificats clients X.509

Les certificats X.509 authentifient les connexions des clients et des appareils à. AWS IoT Les certificats X.509 offrent plusieurs avantages par rapport à d'autres mécanismes d'identification et d'authentification. Les certificats X.509 activent des clés asymétriques à utiliser avec les appareils. Par exemple, vous pouvez graver des clés privées dans un stockage sécurisé sur un appareil afin que le matériel cryptographique sensible ne quitte jamais l'appareil. Les certificats X.509 offrent une authentification du client plus fiable que d'autres méthodes, telles que le nom d'utilisateur et le mot de passe ou les jetons de porteur, car la clé privée ne quitte jamais l'appareil.

AWS IoT authentifie les certificats clients en utilisant le mode d'authentification client du protocole TLS. La prise en charge de TLS est disponible dans de nombreux langages de programmation et systèmes d'exploitation ; ce protocole est couramment utilisée pour le chiffrement des données. Dans le cadre de l'authentification client TLS, AWS IoT demande un certificat client X.509 et valide le statut

du certificat par Compte AWS rapport à un registre de certificats. Il demande ensuite au client de prouver qu'il possède la clé privée correspondant à la clé publique contenue dans le certificat. AWS IoT oblige les clients à envoyer l'extension SNI (Server Name Indication) au protocole TLS (Transport Layer Security). Pour de plus amples informations sur la configuration de l'extension SNI, veuillez consulter Sûreté des transports dans AWS IoT Core.

Pour faciliter une connexion client sécurisée et cohérente au AWS IoT cœur, un certificat client X.509 doit posséder les éléments suivants :

- Enregistré dans AWS IoT Core. Pour de plus amples informations, veuillez consulter Enregistrement d’un certificat client.
- Avoir un état de statut de`ACTIVE`. Pour de plus amples informations, veuillez consulter Activation ou désactivation d’un certificat client.
- Pas encore atteint la date d'expiration du certificat.

Vous pouvez créer des certificats clients qui utilisent Amazon Root CA et vous pouvez utiliser vos propres certificats clients signés par une autre autorité de certification (CA). Pour plus d'informations sur l'utilisation de la AWS IoT console pour créer des certificats utilisant l'autorité de certification Amazon Root, consultezCréation de certificats AWS IoT clients. Pour plus d'informations sur l'utilisation de vos propres certificats X.509, consultez Création de vos propres certificats clients.

La date et l'heure d'expiration des certificats signés par un certificat d'autorité de certification sont définies au moment de la création du certificat. Les certificats X.509 générés par AWS IoT expirent à minuit UTC le 31 décembre 2049 (2049-12-31T 23:59:59 Z).

AWS IoT Device Defender peut effectuer des audits sur votre appareil Compte AWS et sur vos appareils conformément aux meilleures pratiques courantes en matière de sécurité de l'IoT. Cela inclut la gestion des dates d'expiration des certificats X.509 signés par votre autorité de certification ou par l'autorité Amazon Root CA. Pour plus d'informations sur la gestion de la date d'expiration d'un certificat, consultez les sections Expiration du certificat de l'appareil et expiration du certificat CA.

Sur le AWS IoT blog officiel, vous trouverez une analyse plus approfondie de la gestion de la rotation des certificats des appareils et des meilleures pratiques en matière de sécurité dans la section Comment gérer la rotation des certificats des appareils IoT à l'aide de AWS IoT.

Utilisation de certificats client X.509 dans plusieurs Compte AWS s avec enregistrement multi-comptes

L'inscription à plusieurs comptes permet de déplacer des appareils entre vos Compte AWS dans la même région ou dans des régions différentes. Avec cela, vous pouvez enregistrer, tester et configurer un appareil dans un compte de préproduction, puis enregistrer et utiliser le même appareil et le même certificat d'appareil dans un compte de production. Vous pouvez également enregistrer le certificat client sur l'appareil ou les certificats de l'appareil sans qu'une autorité de certification soit enregistrée auprès de celle-ci AWS IoT. Pour plus d'informations, voir Enregistrer un certificat client signé par une autorité de certification (CLI) non enregistrée.

> **Note**
>
> Les certificats utilisés pour l'enregistrement de plusieurs comptes sont pris en charge sur `iot:Data-ATS` les types `iot:Data` (anciens), `iot:Jobs`, et de point de terminaison `iot:CredentialProvider`. Pour plus d'informations sur les points de terminaison des AWS IoT appareils, consultezAWS IoT données de l'appareil et points de terminaison de service.

Les appareils qui utilisent l'enregistrement multi-comptes doivent envoyer l'extension SNI (Server Name Indication) au protocole TLS (Transport Layer Security) et fournir l'adresse complète du point de terminaison `host_name` sur le terrain, lorsqu'ils se connectent à. AWS IoT AWS IoT utilise l'adresse du point de terminaison `host_name` pour acheminer la connexion vers le AWS IoT compte approprié. Les périphériques existants qui n'envoient pas d'adresse de point de terminaison valide dans `host_name` continueront de fonctionner, mais ils ne pourront pas utiliser les fonctionnalités qui nécessitent ces informations. Pour de plus amples informations sur l'extension SNI et pour savoir comment identifier l'adresse du point de terminaison pour le champ `host_name`, veuillez consulter Sûreté des transports dans AWS IoT Core.

Pour utiliser l'enregistrement de plusieurs comptes

1. Vous pouvez enregistrer les certificats d'appareil auprès d'une autorité de certification. Vous pouvez enregistrer l'autorité de certification signataire dans plusieurs comptes en `SNI_ONLY` mode et utiliser cette autorité de certification pour enregistrer le même certificat client sur plusieurs comptes. Pour de plus amples informations, veuillez consulter Enregistrer un certificat CA en mode SNI_ONLY (CLI) - Recommandé.

2. Vous pouvez enregistrer les certificats d'appareil sans autorité de certification. Consultez Enregistrement d'un certificat client signé par une autorité de certification non enregistrée (CLI). L'enregistrement d'une autorité de certification est facultatif. Vous n'êtes pas obligé d'enregistrer l'autorité de certification auprès de laquelle les certificats de l'appareil ont été signés AWS IoT.

Algorithmes de signature de certificats supportés par AWS IoT

AWS IoT prend en charge les algorithmes de signature de certificats suivants :

- SHA256AVEC RSA
- SHA384AVEC RSA
- SHA512AVEC RSA
- SHA256WITHRSAANDMGF1 (RSASSA-PSS)
- SHA384WITHRSAANDMGF1 (RSASSA-PSS)
- SHA512WITHRSAANDMGF1 (RSASSA-PSS)
- DSA_AVEC_ SHA256
- ECDSA-AVEC- SHA256
- ECDSA-AVEC- SHA384
- ECDSA-AVEC- SHA512

Pour plus d'informations sur l'authentification et la sécurité des certificats, consultez la section Qualité de la clé de certificat de l'appareil.

> **Note**
>
> La demande de signature de certificat (CSR) doit inclure une clé publique. La clé peut être soit une clé RSA d'une longueur d'au moins 2 048 bits, soit une clé ECC issue des courbes NIST P-256, NIST P-384 ou NIST P-521. Pour plus d'informations, consultez CreateCertificateFromCsr dans le Guide de référence des API AWS IoT .

Algorithmes clés pris en charge par AWS IoT

Le tableau ci-dessous montre comment les algorithmes clés sont pris en charge :

Algorithme clé	Algorithme de signature de certificat	Version de TLS	Pris en charge ? Oui ou Non
RSA avec une taille de clé d'au moins 2048 bits	Tous	TLS 1.2 TLS 1.3	Oui
ECC NIST P-256/P-384/ P-521	Tous	TLS 1.2 TLS 1.3	Oui
RSA-PSS avec une taille de clé d'au moins 2048 bits	Tous	TLS 1.2	Non
RSA-PSS avec une taille de clé d'au moins 2048 bits	Tous	TLS 1.3	Oui

Pour créer un certificat à l'aide du CreateCertificateFromCSR, vous pouvez utiliser un algorithme de clé pris en charge pour générer une clé publique pour votre CSR. Pour enregistrer votre propre certificat à l'aide de RegisterCertificatenotre autorité de RegisterCertificateWithoutcertification, vous pouvez utiliser un algorithme de clé pris en charge afin de générer une clé publique pour le certificat.

Pour plus d'informations, consultez la section Politiques de sécurité.

Création de certificats AWS IoT clients

AWS IoT fournit des certificats clients signés par l'autorité de certification Amazon Root (CA).

Cette rubrique décrit comment créer un certificat client signé par l'autorité de certification racine Amazon et télécharger les fichiers de certificat. Après avoir créé les fichiers de certificat client, vous devez les installer sur le client.

> **Note**
>
> Chaque certificat client X.509 fourni par AWS IoT contient les attributs d'émetteur et de sujet que vous avez définis au moment de la création du certificat. Les attributs du certificat ne sont immuables qu'une fois le certificat créé.

Vous pouvez utiliser la AWS IoT console ou le AWS CLI pour créer un AWS IoT certificat signé par l'autorité de certification Amazon Root.

Création d'un AWS IoT certificat (console)

Pour créer un AWS IoT certificat à l'aide de la AWS IoT console

1. Connectez-vous à la AWS IoT console AWS Management Console et ouvrez-la.
2. Dans le volet de navigation, choisissez successivement Sécurité, Certificats puis Créer.
3. Choisissez Création d'un certificat en un clic (recommandé) - Créer un certificat.
4. À partir de la page Certificat créé, téléchargez les fichiers du certificat client pour l'objet, la clé publique et privée sur un emplacement sécurisé. Ces certificats générés par ne AWS IoT sont disponibles que pour une utilisation avec AWS IoT les services.

   Si vous avez également besoin du fichier de certificat Amazon Root CA, cette page contient également le lien vers la page où vous pouvez le télécharger.
5. Un certificat client est désormais créé et enregistré auprès d' AWS IoT. Vous devez activer le certificat avant de l'utiliser dans un client.

   Pour activer le certificat client maintenant, choisissez Activer. Si vous ne souhaitez pas activer le certificat maintenant, consultez Activation d'un certificat client (console) la section pour savoir comment activer le certificat ultérieurement.
6. Si vous souhaitez attacher une politique au certificat, choisissez Attacher une Politique.

   Si vous ne souhaitez pas attacher de stratégie maintenant, choisissez Terminé. Vous pouvez attacher une stratégie ultérieurement.

Après avoir terminé la procédure, installez les fichiers de certificat sur le client.

Création d'un AWS IoT certificat (CLI)

AWS CLI Fournit la create-keys-and-certificate commande permettant de créer des certificats clients signés par l'autorité de certification Amazon Root. Toutefois, cette commande ne télécharge pas le fichier de certificat de l'autorité de certification racine Amazon. Vous pouvez télécharger le fichier de certificat de l'autorité de certification racine Amazon à partir de Certificats d'autorité de certification pour l'authentification du serveur.

Cette commande crée des fichiers de clé privée, de clé publique et de certificat X.509, enregistre et active le certificat avec AWS IoT.

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Si vous ne souhaitez pas activer le certificat lorsque vous le créez et l'enregistrez, cette commande crée les fichiers de clé privée, de clé publique et de certificat X.509, enregistre le certificat, mais ne l'active pas. Activation d'un certificat client (CLI) décrit comment activer le certificat ultérieurement.

```
aws iot create-keys-and-certificate \
    --no-set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

Installez les fichiers de certificat sur le client.

Création de vos propres certificats clients

AWS IoT prend en charge les certificats clients signés par n'importe quelle autorité de certification racine ou intermédiaire (CA). AWS IoT utilise des certificats CA pour vérifier la propriété des certificats. Pour utiliser des certificats d'appareil signés par une autorité de certification autre que celle d'Amazon, le certificat de l'autorité de certification doit être enregistré auprès de l'autorité de certification AWS IoT afin que nous puissions vérifier la propriété du certificat de l'appareil.

AWS IoT prend en charge plusieurs méthodes pour apporter vos propres certificats (BYOC) :

- Tout d'abord, enregistrez l'autorité de certification utilisée pour signer les certificats clients, puis enregistrez les certificats clients individuels. Si vous souhaitez enregistrer l'appareil ou le client sur son certificat client lors de sa première connexion AWS IoT (également appelé provisionnement juste à temps), vous devez enregistrer l'autorité de certification signataire auprès AWS IoT de laquelle il se connecte et activer l'enregistrement automatique.
- Si vous ne pouvez pas enregistrer l'autorité de certification signataire, vous pouvez choisir d'enregistrer les certificats clients sans l'autorité de certification. Pour les appareils enregistrés sans

autorité de certification, vous devez présenter une indication du nom du serveur (SNI) lorsque vous les AWS IoT connectez.

> **Note**
>
> Pour enregistrer des certificats clients à l'aide de l'autorité de certification, vous devez enregistrer l'autorité de certification signataire AWS IoT, et non auprès d'une autre autorité de certification CAs dans la hiérarchie.

> **Note**
>
> Un certificat d'autorité de certification peut être enregistré en mode `DEFAULT` que par un seul compte dans une région. Un certificat d'autorité de certification peut être enregistré en mode `SNI_ONLY` que par plusieurs comptes dans une région.

Pour plus d'informations sur l'utilisation des certificats X.509 pour la prise en charge de plusieurs appareils, veuillez consulter Mise en service des appareils pour prendre connaissance des différentes options de gestion et de mise en service des certificats prises en charge par AWS IoT .

Rubriques

- Gestion de vos certificats d’autorité de certification
- Création d’un certificat client à l'aide de votre certificat d’autorité de certification

Gestion de vos certificats d’autorité de certification

Cette section décrit les tâches courantes de gestion de vos propres certificats d'autorité de certification.

Vous pouvez enregistrer votre autorité de certification (CA) auprès de laquelle AWS IoT vous utilisez des certificats clients signés par une autorité de AWS IoT certification non reconnue.

Si vous souhaitez que les clients enregistrent automatiquement leurs certificats clients AWS IoT lors de leur première connexion, l'autorité de certification qui a signé les certificats clients doit être enregistrée auprès de l'autorité de certification AWS IoT. Sinon, vous n'avez pas besoin d'enregistrer le certificat de l'autorité de certification qui a signé les certificats clients.

> **Note**
>
> Un certificat d'autorité de certification peut être enregistré en mode `DEFAULT` que par un seul compte dans une région. Un certificat d'autorité de certification peut être enregistré en mode `SNI_ONLY` que par plusieurs comptes dans une région.

Rubriques :

- Création d’un certificat d’autorité de certification
- Enregistrement de votre certificat d’autorité de certification
- Désactivation d'un certificat d'autorité de certification

Création d’un certificat d’autorité de certification

Si vous ne disposez pas de certificat CA, vous pouvez utiliser les outils OpenSSL v1.1.1i pour en créer un.

> **Note**
>
> Vous ne pouvez pas exécuter cette procédure dans la AWS IoT console.

Pour créer un certificat CA à l'aide des outils OpenSSL v1.1.1i

1. Générez une paire de clés.

```
openssl genrsa -out root_CA_key_filename.key 2048
```

2. Utilisez la clé privée de la paire de clés pour générer un certificat CA.

```
openssl req -x509 -new -nodes \
    -key root_CA_key_filename.key \
    -sha256 -days 1024 \
    -out root_CA_cert_filename.pem
```

Enregistrement de votre certificat d’autorité de certification

Ces procédures décrivent comment enregistrer un certificat auprès d'une autorité de certification (CA) autre que l'autorité de certification d'Amazon. AWS IoT Core utilise des certificats CA pour vérifier la propriété des certificats. Pour utiliser des certificats d'appareil signés par une autorité de certification autre que celle d'Amazon, vous devez enregistrer le certificat auprès de l'autorité de certification AWS IoT Core afin qu'elle puisse vérifier la propriété du certificat de l'appareil.

Enregistrement d’un certificat d'autorité de certification (console)

> **Note**
>
> Pour enregistrer un certificat CA dans la console, commencez par enregistrer le certificat CA dans la console. Vous pouvez enregistrer votre autorité de certification en mode multi-comptes, sans avoir à fournir de certificat de vérification ni d’accéder à la clé privée. Une autorité de certification peut être enregistrée en mode multi-comptes par plusieurs Comptes AWS dans un même Région AWS. Vous pouvez enregistrer votre autorité de certification en mode compte unique en fournissant un certificat de vérification et une preuve de propriété de la clé privée de l'autorité de certification.

Enregistrement d’un certificat d'autorité de certification (CLI)

Vous pouvez enregistrer un certificat CA en mode `DEFAULT` ou en mode `SNI_ONLY`. Une autorité de certification peut être enregistrée en `DEFAULT` mode une Compte AWS par une Région AWS. Une autorité de certification peut être enregistrée en `SNI_ONLY` mode par plusieurs Comptes AWS dans le même mode Région AWS. Pour en savoir plus sur les certificats CA, veuillez consulter Modecertificat.

> **Note**
>
> Nous vous recommandons d'enregistrer une autorité de certification en mode `SNI_ONLY`. Vous n'avez pas besoin de fournir de certificat de vérification ni d'accès à la clé privée, et vous pouvez enregistrer l'autorité de certification par plusieurs Comptes AWS fois Région AWS.

Enregistrer un certificat CA en mode SNI_ONLY (CLI) - Recommandé

Prérequis

Vérifiez que les éléments suivants sont disponibles sur votre ordinateur avant de continuer :

- Le fichier de certificat de l'autorité de certification racine (référencé dans l'exemple suivant sous le nom *root_CA_cert_filename.pem*)
- OpenSSL v1.1.1i ou version ultérieure

Pour enregistrer un certificat CA en **SNI_ONLY** mode à l'aide du AWS CLI

1. Enregistrez le certificat CA auprès de AWS IoT. À l'aide de la commande register-ca-certificate, entrez le nom du fichier du certificat CA. Pour plus d'informations, consultez register-ca-certificate dans la Référence des commandes de l'AWS CLI .

   ```
   aws iot register-ca-certificate \
       --ca-certificate file://root_CA_cert_filename.pem \
       --certificate-mode SNI_ONLY
   ```

   En cas de succès, cette commande renvoie le*certificateId*.
2. À ce stade, le certificat CA a été enregistré AWS IoT mais il est inactif. Le certificat de l'autorité de certification doit être actif avant que vous puissiez enregistrer les certificats clients qu'il a signés.

   Cette étape active le certificat de l'autorité de certification.

   Pour activer le certificat CA, utilisez la commande update-certificate comme suit. Pour plus d'informations, consultez mettre à jour-le certificat dans la AWS CLI Référence des commandes.

   ```
   aws iot update-ca-certificate \
       --certificate-id certificateId \
       --new-status ACTIVE
   ```

Utilisez la commande describe-ca-certificate pour voir le statut du certificat CA. Pour plus d'informations, consultez describe-ca-certificate dans la Référence des commandes de l'AWS CLI .

Enregistrement d'un certificat CA en **DEFAULT** mode (CLI)

Prérequis

Vérifiez que les éléments suivants sont disponibles sur votre ordinateur avant de continuer :

- Le fichier de certificat de l'autorité de certification racine (référencé dans l'exemple suivant sous le nom *root_CA_cert_filename.pem*)
- Le fichier de certificat de l'autorité de certification racine (référencé dans l'exemple suivant sous *root_CA_key_filename.key*)
- OpenSSL v1.1.1i ou version ultérieure

Pour enregistrer un certificat CA en **DEFAULT** mode à l'aide du AWS CLI

1. Pour obtenir un code d'enregistrement auprès de AWS IoT, utilisezget-registration-code. Enregistrez le `registrationCode` renvoyé à utiliser en tant que `Common Name` du certificat de vérification de clé privée. Pour plus d'informations, consultez get-registration-code dans la Référence des commandes de l'AWS CLI .

   ```
   aws iot get-registration-code
   ```

2. Générez une paire de clés pour le certificat de vérification de clé privée :

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

3. Créez une demande de signature de certificat (CSR) pour le certificat de vérification de clé privée. Dans le champ `Common Name` du certificat, indiquez la valeur `registrationCode` renvoyée par get-registration-code.

   ```
   openssl req -new \
       -key verification_cert_key_filename.key \
       -out verification_cert_csr_filename.csr
   ```

   Vous êtes invité à entrer certaines informations, notamment le `Common Name` du certificat.

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:
       State or Province Name (full name) []:
       Locality Name (for example, city) []:
   ```

```
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

4. Utilisez la CSR pour créer un certificat de vérification de clé privée :

```
openssl x509 -req \
    -in verification_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

5. Enregistrez le certificat CA auprès de AWS IoT. Transmettez le nom de fichier de certificat CA et le nom du fichier de certificat de vérification de clé privée à la commande register-ca-certificate, comme suit. Pour plus d'informations, consultez register-ca-certificate dans la Référence des commandes de l'AWS CLI .

```
aws iot register-ca-certificate \
    --ca-certificate file://root_CA_cert_filename.pem \
    --verification-cert file://verification_cert_filename.pem
```

   Cette commande renvoie le*certificateId*, en cas de succès.

6. À ce stade, le certificat CA a été enregistré AWS IoT mais n'est pas actif. Le certificat CA doit être actif avant que vous puissiez enregistrer les certificats clients qu'il a signés.

   Cette étape active le certificat de l'autorité de certification.

   Pour activer le certificat CA, utilisez la commande update-certificate comme suit. Pour plus d'informations, consultez mettre à jour-le certificat dans la AWS CLI Référence des commandes.

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

Utilisez la commande describe-ca-certificate pour voir le statut du certificat CA. Pour plus d'informations, consultez describe-ca-certificate dans la Référence des commandes de l'AWS CLI .

Créez un certificat de vérification CA pour enregistrer le certificat CA dans la console

> **Note**
>
> Cette procédure est uniquement destinée à être utilisée si vous enregistrez un certificat CA depuis la AWS IoT console.
> Si vous n'avez pas accédé à cette procédure depuis la AWS IoT console, lancez le processus d'enregistrement du certificat CA dans la console à l'adresse Enregistrer le certificat CA.

Vérifiez que les éléments suivants sont disponibles sur le même ordinateur avant de continuer :

- Le fichier de certificat de l'autorité de certification racine (référencé dans l'exemple suivant sous le nom *root_CA_cert_filename.pem*)
- Le fichier de certificat de l'autorité de certification racine (référencé dans l'exemple suivant sous *root_CA_key_filename.key*)
- OpenSSL v1.1.1i ou version ultérieure

Pour utiliser l'interface de ligne de commande pour créer un certificat de vérification CA afin d'enregistrer votre certificat CA dans la console

1. *verification_cert_key_filename.key*Remplacez-le par le nom du fichier clé du certificat de vérification que vous souhaitez créer (par exemple,**verification_cert.key**). Exécutez ensuite cette commande pour générer une paire de clés pour le certificat de vérification de clé privée :

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

2. Remplacez *verification_cert_key_filename.key* par le nom du fichier de clé que vous avez créé à l'étape 1.

   Remplacez *verification_cert_csr_filename.csr* par le nom du fichier de demande de signature du certificat (CSR) que vous souhaitez créer. Par exemple, **verification_cert.csr**.

   Exécutez cette commande pour créer le fichier CSR.

```
openssl req -new \
    -key verification_cert_key_filename.key \
    -out verification_cert_csr_filename.csr
```

La commande vous invite à saisir des informations supplémentaires qui seront expliquées plus loin.

3. Dans la AWS IoT console, dans le conteneur du certificat de vérification, copiez le code d'enregistrement.
4. Les informations que la commande openssl vous demande de saisir sont présentées dans l'exemple suivant. À l'exception du champ `Common Name`, vous pouvez saisir vos propres valeurs ou les laisser vides.

   Dans le champ `Common Name`, collez le code d'enregistrement que vous avez copié à l'étape précédente.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

   Une fois que vous avez terminé, la commande crée le fichier CSR.

5. Remplacez la *verification_cert_csr_filename.csr* par la *verification_cert_csr_filename.csr* que vous avez utilisée à l'étape précédente.

Remplacez *root_CA_cert_filename.pem* par le nom de fichier du certificat CA que vous souhaitez enregistrer.

Remplacez *root_CA_key_filename.key* par le nom du fichier de clé privée du certificat CA.

Remplacez *verification_cert_filename.pem* par le nom de fichier du certificat de vérification que vous souhaitez créer. Par exemple, **verification_cert.pem**.

```
openssl x509 -req \
    -in verification_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

6. Une fois la commande OpenSSL terminée, vous devriez avoir ces fichiers prêts à être utilisés lorsque vous revenez à la console.

   - Votre fichier de certificat CA (*root_CA_cert_filename.pem* utilisé dans la commande précédente)
   - Le certificat de vérification que vous avez créé à l'étape précédente (*verification_cert_filename.pem*utilisé dans la commande précédente)

Désactivation d'un certificat d'autorité de certification

Lorsqu'un certificat d'autorité de certification (CA) est activé pour l'enregistrement automatique des certificats clients, AWS IoT vérifie le certificat de l'autorité de certification pour s'assurer que l'autorité de certification l'est`ACTIVE`. Si le certificat CA l'est`INACTIVE`, AWS IoT cela n'autorise pas l'enregistrement du certificat client.

En définissant le certificat de l'autorité de certification sur `INACTIVE`, vous empêchez l'enregistrement automatique de tout nouveau certificat client émis par l'autorité de certification.

> **Note**
>
> Les certificats clients enregistrés qui ont été signés par le certificat d'autorité de certification compromis continuent de fonctionner jusqu'à ce que vous les révoquiez explicitement.

Désactivation d'un certificat d'autorité de certification (console)

Pour désactiver un certificat d'autorité de certification à l'aide de la console AWS IoT

1. Connectez-vous à la AWS IoT console AWS Management Console et ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Secure, choisissez CAs.
3. Dans la liste des autorités de certification, recherchez celle que vous souhaitez désactiver et choisissez l'icône de points de suspension pour ouvrir le menu d'options.
4. Dans le menu d'options, choisissez Désactiver.

L'autorité de certification doit apparaître comme Inactive dans la liste.

> **Note**
>
> La AWS IoT console ne permet pas de répertorier les certificats signés par l'autorité de certification que vous avez désactivée. Pour connaître l'option d' AWS CLI permettant de répertorier ces certificats, veuillez consulter Désactivation d'un certificat d'autorité de certification (CLI).

Désactivation d'un certificat d'autorité de certification (CLI)

AWS CLI Fournit la update-ca-certificatecommande permettant de désactiver un certificat CA.

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status INACTIVE
```

Utilisez la commande list-certificates-by-ca pour obtenir la liste de tous les certificats clients enregistrés qui ont été signés par l'autorité de certification spécifiée. Pour chaque certificat client signé par le certificat d'autorité de certification spécifié, utilisez la commande update-certificate pour révoquer le certificat client afin d'empêcher son utilisation.

Utilisez la commande describe-ca-certificate pour voir le statut du certificat d'autorité de certification.

Création d'un certificat client à l'aide de votre certificat d'autorité de certification

Vous pouvez utiliser votre propre autorité de certification pour créer des certificats clients. Le certificat client doit être enregistré AWS IoT avant de pouvoir être utilisé. Pour de plus amples informations

sur les options d'enregistrement pour vos certificats clients, veuillez consulter Enregistrement d'un certificat client.

Création d'un certificat client (CLI)

> **Note**
>
> Vous ne pouvez pas exécuter cette procédure dans la AWS IoT console.

Pour créer un certificat client à l'aide du AWS CLI

1. Générez une paire de clés.

```
openssl genrsa -out device_cert_key_filename.key 2048
```

2. Créez une demande de signature de certificat (CSR) pour le certificat client.

```
openssl req -new \
    -key device_cert_key_filename.key \
    -out device_cert_csr_filename.csr
```

Vous êtes invité à entrer certaines informations, comme illustré ici :

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
    State or Province Name (full name) []:
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
```

```
    An optional company name []:
```

3. Créez un certificat client à partir de la CSR.

```
openssl x509 -req \
    -in device_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out device_cert_filename.pem \
    -days 500 -sha256
```

À ce stade, le certificat client a été créé, mais il n'a pas encore été enregistré auprès de celui-ci AWS IoT. Pour de plus amples informations sur la façon et le moment d'enregistrer le certificat client, veuillez consulter Enregistrement d’un certificat client.

Enregistrement d’un certificat client

Les certificats clients doivent être enregistrés AWS IoT pour permettre les communications entre le client et AWS IoT. Vous pouvez enregistrer chaque certificat client manuellement ou configurer les certificats client pour qu'ils s'enregistrent automatiquement lorsque le client se connecte AWS IoT pour la première fois.

Si vous souhaitez que vos clients et appareils enregistrent leurs certificats clients lors de leur première connexion, vous devez utiliser Enregistrement de votre certificat d’autorité de certification pour signer le certificat client auprès d’ AWS IoT dans les régions où vous souhaitez l'utiliser. L'autorité de certification Amazon Root est automatiquement enregistrée auprès de AWS IoT.

Les certificats clients peuvent être partagés par Comptes AWS et par régions. Les procédures décrites dans ces rubriques doivent être effectuées dans chaque compte et chaque région où vous souhaitez utiliser le certificat client. L'enregistrement d'un certificat client dans un compte ou une région n'est pas automatiquement reconnu par un autre compte ou une autre région.

> **Note**
>
> Les clients qui utilisent le protocole TLS (Transport Layer Security) pour se connecter à AWS IoT doivent prendre en charge l'extension SNI (Server Name Indication) de TLS. Pour de plus amples informations, veuillez consulter Sûreté des transports dans AWS IoT Core.

Rubriques

- Enregistrement manuel d’un certificat client
- Enregistrer un certificat client lorsque le client se connecte à l' AWS IoT just-in-time enregistrement (JITR)

Enregistrement manuel d’un certificat client

Vous pouvez enregistrer un certificat client manuellement à l'aide de la AWS IoT console et AWS CLI.

La procédure d'enregistrement à utiliser dépend du fait que le certificat sera partagé ou non par Compte AWS les régions. L'enregistrement d'un certificat client dans un compte ou une région n'est pas automatiquement reconnu par un autre compte ou une autre région.

Les procédures décrites dans cette rubrique doivent être effectuées dans chaque compte et chaque région où vous souhaitez utiliser le certificat client. Les certificats clients peuvent être partagés par Compte AWS s et par régions.

Enregistrement d’un certificat client signé par une autorité de certification enregistrée (console)

> **Note**
>
> Avant d'exécuter cette procédure, assurez-vous que vous disposez du fichier .pem du certificat client et que le certificat client a été signé par une autorité de certification auprès de laquelle vous vous êtes enregistré. AWS IoT

Pour enregistrer un certificat existant à AWS IoT l'aide de la console

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le panneau de navigation, sous la section Gérer, choisissez Sécurité, puis Certificats.
3. Sur la page Certificats de la boîte de dialogue Certificats, choisissez Ajouter un certificat, puis Enregistrer les certificats.
4. Sur la page Enregistrer le certificat de la boîte de dialogue Certificats à télécharger, procédez comme suit :

   - Choisissez CA est enregistré auprès de AWS IoT.
   - Dans Choisir un certificat CA, sélectionnez votre autorité de certification.

    - Choisissez Enregistrer une nouvelle autorité de certification pour enregistrer une nouvelle autorité de certification qui n'est pas enregistrée auprès de celle-ci AWS IoT.
    - Laissez le champ Choisir un certificat CA vide si l'autorité de certification Amazon Root est votre autorité de certification.
- Sélectionnez jusqu'à 10 certificats à télécharger et à enregistrer AWS IoT.
    - Utilisez les fichiers de certificat que vous avez créés dans Création de certificats AWS IoT clients etCréation d’un certificat client à l'aide de votre certificat d’autorité de certification.
- Choisissez Activer ou Désactiver. Si vous choisissez Désactiver, Activation ou désactivation d’un certificat client explique comment activer votre certificat après l'avoir enregistré.
- Choisissez Register (S'inscrire).

Sur la page Certificats de la boîte de dialogue Certificats, vos certificats enregistrés apparaissent désormais.

Enregistrement d’un certificat client signé par une autorité de certification non enregistrée (console)

> **Note**
>
> Avant d'effectuer cette procédure, assurez-vous de disposer du fichier .pem du certificat client.

Pour enregistrer un certificat existant à AWS IoT l'aide de la console

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez successivement Sécurité, Certificats et Créer.
3. Dans Créer un certificat, recherchez l'entrée Utiliser mon certificat, puis choisissez Commencer.
4. Dans Sélectionner une autorité de certification, choisissez Suivant.
5. Dans Enregistrer des certificats d'appareils existants, choisissez Sélectionner les certificats et sélectionnez jusqu'à 10 fichiers de certificats à enregistrer.
6. Après avoir fermé la boîte de dialogue du fichier, indiquez si vous souhaitez activer ou révoquer les certificats clients lorsque vous les enregistrez.

   Si vous n'activez pas un certificat lors de son enregistrement, Activation d’un certificat client (console) décrit comment l'activer ultérieurement.

Si un certificat est révoqué lors de son enregistrement, il ne peut pas être activé ultérieurement.

Après avoir sélectionné les fichiers de certificat à enregistrer et sélectionné les actions à effectuer après l'enregistrement, sélectionnez Enregistrer les certificats.

Les certificats clients enregistrés apparaissent dans la liste des certificats.

Enregistrement d'un certificat client signé par une autorité de certification enregistrée (CLI)

> **Note**
>
> Avant d'effectuer cette procédure, assurez-vous de disposer du fichier .pem de l'autorité de certification et du fichier .pem du certificat client. Le certificat client doit être signé par une autorité de certification (CA) auprès de laquelle vous vous êtes enregistré AWS IoT.

Utilisez la commande register-certificate pour enregistrer un certificat client sans l'activer.

```
aws iot register-certificate \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

Le certificat client est enregistré auprès de AWS IoT, mais il n'est pas encore actif. Veuillez consulter Activation d'un certificat client (CLI) pour de plus amples informations sur la façon de l'activer ultérieurement.

Vous pouvez également activer le certificat client lorsque vous l'enregistrez à l'aide de cette commande.

```
aws iot register-certificate \
    --set-as-active \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

Pour plus d'informations sur l'activation du certificat afin qu'il puisse être utilisé pour se connecter à AWS IoT, voir Activation ou désactivation d'un certificat client

Enregistrement d’un certificat client signé par une autorité de certification non enregistrée (CLI)

> **Note**
>
> Avant d'effectuer cette procédure, assurez-vous de disposer du fichier .pem du certificat.

Utilisez la commande register-certificate-without-ca pour enregistrer un certificat client sans l’activer.

```
aws iot register-certificate-without-ca \
    --certificate-pem file://device_cert_filename.pem
```

Le certificat client est enregistré auprès de AWS IoT, mais il n'est pas encore actif. Veuillez consulter Activation d’un certificat client (CLI) pour de plus amples informations sur la façon de l'activer ultérieurement.

Vous pouvez également activer le certificat client lorsque vous l'enregistrez à l'aide de cette commande.

```
aws iot register-certificate-without-ca \
    --status ACTIVE \
    --certificate-pem file://device_cert_filename.pem
```

Pour plus d'informations sur l'activation du certificat afin qu'il puisse être utilisé pour se connecter AWS IoT, consultezActivation ou désactivation d’un certificat client.

Enregistrer un certificat client lorsque le client se connecte à l' AWS IoT just-in-time enregistrement (JITR)

Vous pouvez configurer un certificat CA pour permettre aux certificats clients qu'il a signés de s'enregistrer AWS IoT automatiquement lors de la première connexion du client AWS IoT.

Pour enregistrer des certificats client lorsqu'un client se connecte AWS IoT pour la première fois, vous devez activer le certificat CA pour l'enregistrement automatique et configurer la première connexion du client afin de fournir les certificats requis.

Configuration d’un certificat d'autorité de certification pour la prise en charge de l'enregistrement automatique (console)

Pour configurer un certificat CA afin de prendre en charge l'enregistrement automatique des certificats clients à l'aide de la AWS IoT console

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Secure, choisissez CAs.
3. Dans la liste des autorités de certification, recherchez celle pour laquelle vous souhaitez activer l'enregistrement automatique et ouvrez le menu d'options à l'aide de l'icône représentant des points de suspension.
4. Dans le menu d'options, choisissez Activer l'enregistrement automatique.

> **Note**
>
> Le statut d'enregistrement automatique n'apparaît pas dans la liste des autorités de certification. Pour voir le statut d'enregistrement automatique d'une autorité de certification, vous devez ouvrir la page Détails de l'autorité de certification.

Configuration d’un certificat d'autorité de certification pour la prise en charge de l'enregistrement automatique (CLI)

Si vous avez déjà enregistré votre certificat CA auprès de AWS IoT, utilisez la update-ca-certificatecommande pour définir `autoRegistrationStatus` le certificat CA sur`ENABLE`.

```
aws iot update-ca-certificate \
--certificate-id caCertificateId \
--new-auto-registration-status ENABLE
```

Si vous souhaitez activer `autoRegistrationStatus` lors de l'enregistrement du certificat d'autorité de certification, utilisez la commande register-ca-certificate.

```
aws iot register-ca-certificate \
--allow-auto-registration  \
--ca-certificate file://root_CA_cert_filename.pem \
--verification-cert file://verification_cert_filename.pem
```

Utilisez la commande describe-ca-certificate pour voir le statut du certificat d'autorité de certification.

Configuration de la première connexion par un client pour l'enregistrement automatique

Lorsqu'un client tente de se connecter AWS IoT pour la première fois, le certificat client signé par votre certificat CA doit être présent sur le client lors de la prise de contact TLS (Transport Layer Security).

Lorsque le client se connecte à AWS IoT, utilisez le certificat client que vous avez créé dans Créer des certificats AWS IoT clients ou Créez vos propres certificats clients. AWS IoT reconnaît le certificat CA en tant que certificat CA enregistré, enregistre le certificat client et définit son statut sur`PENDING_ACTIVATION`. Cela signifie que le certificat client a été enregistré automatiquement et qu'il est en attente d'activation. L'état du certificat client doit être `ACTIVE` avant de pouvoir être utilisé pour la connexion à AWS IoT. Consultez Activation ou désactivation d’un certificat client pour plus d'informations sur l'activation d'un certificat client.

> **Note**
>
> Vous pouvez approvisionner des appareils à l'aide de la fonction AWS IoT Core just-in-time d'enregistrement (JITR) sans avoir à envoyer l'intégralité de la chaîne de confiance lors de la première connexion des appareils à. AWS IoT Core La présentation du certificat CA est facultative, mais l'appareil doit envoyer l'extension SNI (Server Name Indication) lors de la connexion.

Lors de l'enregistrement AWS IoT automatique d'un certificat ou lorsqu'un client présente un certificat avec le `PENDING_ACTIVATION` statut, AWS IoT publie un message sur la rubrique MQTT suivante :

`$aws/events/certificates/registered/`*`caCertificateId`*

Où *`caCertificateId`* est l'ID du certificat d’autorité de certification ayant émis le certificat client.

Le message publié sur cette rubrique a la structure suivante :

```
{
        "certificateId": "certificateId",
        "caCertificateId": "caCertificateId",
        "timestamp": timestamp,
        "certificateStatus": "PENDING_ACTIVATION",
```

```
        "awsAccountId": "awsAccountId",
        "certificateRegistrationTimestamp": "certificateRegistrationTimestamp"
}
```

Vous pouvez créer une règle qui écoute cette rubrique et effectue certaines actions. Nous vous recommandons de créer une règle Lambda qui vérifie que le certificat client ne figure pas sur une liste de révocation de certificats (CRL), active le certificat, puis crée et attache une politique au certificat. La stratégie détermine les ressources auxquelles le client peut accéder. Si la politique que vous créez nécessite l'ID client des appareils connectés, vous pouvez utiliser la fonction clientid () de la règle pour récupérer l'ID client. Voici un exemple de définition de règle :

```
SELECT *,
   clientid() as clientid
from $aws/events/certificates/registered/caCertificateId
```

Dans cet exemple, la règle s'abonne à la rubrique JITR `$aws/events/certificates/registered/caCertificateID` et utilise la fonction clientid () pour récupérer l'identifiant du client. La règle ajoute ensuite l'ID client à la charge utile JITR. Pour plus d'informations sur la fonction clientid () de la règle, consultez clientid ().

Pour plus d'informations sur la création d'une règle Lambda qui écoute le `$aws/events/certificates/registered/caCertificateID` sujet et exécute ces actions, consultez la section just-in-time Enregistrement des certificats clients sur. AWS IoT

Si une erreur ou une exception survient lors de l'enregistrement automatique des certificats clients, AWS IoT envoie des événements ou des messages à vos journaux de CloudWatch connexion. Pour plus d'informations sur la configuration des journaux de votre compte, consultez la CloudWatch documentation Amazon.

Gérer les certificats clients

AWS IoT fournit des fonctionnalités vous permettant de gérer les certificats clients.

Dans cette rubrique :

- Activation ou désactivation d’un certificat client
- Attacher un objet ou une stratégie à un certificat client
- Révocation d’un certificat client
- Transférer un certificat vers un autre compte

Activation ou désactivation d’un certificat client

AWS IoT vérifie qu'un certificat client est actif lorsqu'il authentifie une connexion.

Vous pouvez créer et enregistrer des certificats clients sans les activer afin qu'ils ne puissent pas être utilisés tant que vous ne le souhaitez pas. Vous pouvez également désactiver temporairement des certificats clients actifs. Enfin, vous pouvez révoquer des certificats clients pour empêcher toute future utilisation.

Activation d’un certificat client (console)

Pour activer un certificat client à l'aide de la AWS IoT console

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.
3. Dans la liste des certificats, recherchez le certificat que vous souhaitez activer et ouvrez le menu d'options à l'aide de l'icône représentant des points de suspension.
4. Dans le menu d'options, choisissez Activer.

Le certificat doit apparaître comme Active dans la liste des certificats.

Désactivation d’un certificat client (console)

Pour désactiver un certificat client à l'aide de la console AWS IoT

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.
3. Dans la liste des certificats, recherchez le certificat que vous souhaitez désactiver et ouvrez le menu d'options à l'aide de l'icône représentant des points de suspension.
4. Dans le menu d'options, choisissez Désactiver.

Le certificat doit apparaître comme Inactive dans la liste des certificats.

Activation d’un certificat client (CLI)

AWS CLI Fournit la update-certificatecommande permettant d'activer un certificat.

```
aws iot update-certificate \
```

```
    --certificate-id certificateId \
    --new-status ACTIVE
```

Si la commande aboutit, le statut du certificat devient `ACTIVE`. Exécutez describe-certificate pour voir le statut du certificat.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Désactivation d'un certificat client (CLI)

AWS CLI Fournit la update-certificatecommande permettant de désactiver un certificat.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status INACTIVE
```

Si la commande aboutit, le statut du certificat devient `INACTIVE`. Exécutez describe-certificate pour voir le statut du certificat.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Attacher un objet ou une stratégie à un certificat client

Lorsque vous créez et enregistrez un certificat distinct d'un AWS IoT objet, il ne comporte aucune politique autorisant AWS IoT des opérations et ne sera associé à aucun AWS IoT objet objet. Cette section décrit comment ajouter ces relations à un certificat enregistré.

> **Important**
>
> Pour effectuer ces procédures, vous devez avoir déjà créé l'objet ou la stratégie que vous souhaitez attacher au certificat.

Le certificat authentifie un appareil AWS IoT afin qu'il puisse se connecter. L'attachement du certificat à une ressource d'objet établit la relation entre l'appareil (par le biais du certificat) et la ressource de l'objet. Pour autoriser l'appareil à effectuer AWS IoT des actions, telles que l'autoriser à se connecter et à publier des messages, une politique appropriée doit être attachée au certificat de l'appareil.

Attacher un objet à un certificat client (console)

Vous aurez besoin du nom de l'objet d'objet pour réaliser cette procédure.

Pour attacher un objet d'objet à un certificat enregistré

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.
3. Dans la liste des certificats, recherchez le certificat auquel vous souhaitez attacher une stratégie, ouvrez le menu d'options du certificat en choisissant l'icône représentant trois points de suspension, puis choisissez Attacher un objet.
4. Dans la fenêtre contextuelle, recherchez le nom de l'objet à attacher au certificat, cochez sa case et choisissez Attacher..

L'objet d'objet doit désormais apparaître dans la liste des objets sur la page de détails du certificat.

Attacher une stratégie à un certificat client (console)

Vous aurez besoin du nom de l'objet de stratégie pour réaliser cette procédure.

Pour attacher un objet de stratégie à un certificat enregistré

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.
3. Dans la liste des certificats, recherchez le certificat auquel vous souhaitez attacher une stratégie, ouvrez le menu d'options du certificat en choisissant l'icône représentant trois points de suspension, puis choisissez Attacher une stratégie.
4. Dans la fenêtre contextuelle, recherchez le nom de la stratégie à attacher au certificat, cochez sa case et choisissez Attacher.

L'objet de stratégie doit désormais apparaître dans la liste des stratégies de la page de détails du certificat.

Attacher un objet à un certificat client (interface de ligne de commande)

AWS CLI Fournit la attach-thing-principalcommande permettant d'associer un objet à un certificat.

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

```
    --thing-name thingName
```

Attacher une stratégie à un certificat client (interface de ligne de commande)

AWS CLI Fournit la attach-policycommande permettant d'associer un objet de politique à un certificat.

```
aws iot attach-policy \
    --target certificateArn \
    --policy-name policyName
```

Révocation d'un certificat client

Si vous détectez une activité suspecte sur un certificat client enregistré, vous pouvez révoquer celui-ci afin qu'il ne puisse plus être utilisé.

> **Note**
>
> Une fois qu'un certificat est révoqué, son statut ne peut pas être modifié. En d'autres termes, le statut du certificat ne peut pas être `Active` remplacé par un autre statut.

Révocation d'un certificat client (console)

Pour révoquer un certificat client à l'aide de la console AWS IoT

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.
3. Dans la liste des certificats, recherchez le certificat que vous souhaitez révoquer et ouvrez le menu d'options à l'aide de l'icône représentant des points de suspension.
4. Dans le menu d'options, choisissez Révoquer.

Si la révocation aboutit, il s'affichera comme Revoked (Révoqué) dans la liste des certificats.

Révocation d'un certificat client (CLI)

AWS CLI Fournit la update-certificatecommande permettant de révoquer un certificat.

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status REVOKED
```

Si la commande aboutit, le statut du certificat devient `REVOKED`. Exécutez describe-certificate pour voir le statut du certificat.

```
aws iot describe-certificate \
    --certificate-id certificateId
```

Transférer un certificat vers un autre compte

Les certificats X.509 appartenant à l'un Compte AWS peuvent être transférés vers un autre Compte AWS.

Pour transférer un certificat X.509 de l'un Compte AWS à l'autre

1. the section called “Commencer un transfert de certificat”

   Le certificat doit être désactivé et détaché de toutes les politiques et autres éléments avant de lancer le transfert.
2. the section called “Accepter ou refuser un transfert de certificat”

   Le compte destinataire doit explicitement accepter ou rejeter le certificat transféré. Une fois que le compte destinataire a accepté le certificat, celui-ci doit être activé avant utilisation.
3. the section called “Annuler un transfert de certificat”

   Le compte d'origine peut annuler un transfert si le certificat n'a pas été accepté.

Commencer un transfert de certificat

Vous pouvez commencer à transférer un certificat vers un autre Compte AWS en utilisant la AWS IoT console ou le AWS CLI.

Commencer un transfert de certificat (console)

Pour effectuer cette procédure, vous avez besoin de l'ID du certificat que vous souhaitez transférer.

Effectuez cette procédure depuis le compte avec le certificat à transférer.

Pour commencer à transférer un certificat vers un autre Compte AWS

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.

Choisissez le certificat avec le statut actif ou inactif que vous souhaitez transférer et ouvrez sa page détails.

3. Sur la page Détails du certificat, dans le menu Actions, si l'option Désactiver est disponible, choisissez l'option Désactiver pour désactiver le certificat.
4. Sur la page Détails du certificat, dans le menu de gauche, sélectionnez Politiques.
5. Sur la page Stratégie du certificat, si des stratégies sont associées au certificat, détachez-les en ouvrant le menu des options de la stratégie et en choisissant Détacher.

   Le certificat ne doit contenir aucune politique attachée avant de continuer.
6. Sur la page Politiques du certificat, dans le menu de gauche, sélectionnez Objets.
7. Sur la page Objets du certificat, si des éléments sont associés au certificat, détachez-les en ouvrant le menu des options de l'objet et en choisissant Détacher.

   Le certificat ne doit contenir aucun objet attaché avant de continuer.
8. Sur la page Objets du certificat, dans le menu de gauche, sélectionnez Détails.
9. Sur la page Détails du certificat, dans le menu Actions, choisissez Démarrer le transfert pour ouvrir la boîte de dialogue Démarrer le transfert.
10. Dans la boîte de dialogue Démarrer le transfert, entrez le Compte AWS numéro du compte qui recevra le certificat et un court message facultatif.
11. Choisissez Démarrer le transfert pour transférer le certificat.

La console doit afficher un message qui indique la réussite ou l'échec du transfert. Si le transfert a été lancé, le statut du certificat passe à Transféré.

Commencer un transfert de certificat (CLI)

Pour effectuer cette procédure, vous aurez besoin du *certificateId* et *certificateArn* du certificat que vous souhaitez transférer.

Effectuez cette procédure depuis le compte avec le certificat à transférer.

Pour commencer à transférer un certificat vers un autre compte AWS

1. Utilisez la commande update-certificate pour désactiver le certificat.

   ```
   aws iot update-certificate --certificate-id certificateId --new-status INACTIVE
   ```

2. Détachez toutes les politiques.

    1. Utilisez la commande list-attached-policies pour répertorier les politiques associées au certificat.

    ```
    aws iot list-attached-policies --target certificateArn
    ```

    2. Pour chaque stratégie attachée, utilisez la commande detach-policy pour la détacher.

    ```
    aws iot detach-policy --target certificateArn --policy-name policy-name
    ```

3. Répertoriez tous les objets.

    1. Utilisez la commande list-principal-things pour répertorier les objets associés au certificat.

    ```
    aws iot list-principal-things --principal certificateArn
    ```

    2. Pour chaque objet attaché, utilisez la commande detach-thing-principal pour le détacher.

    ```
    aws iot detach-thing-principal --principal certificateArn --thing-name thing-name
    ```

4. Utilisez la commande transfer-certificate pour démarrer le transfert du certificat.

```
aws iot transfer-certificate --certificate-id certificateId --target-aws-
account account-id
```

Accepter ou refuser un transfert de certificat

Vous pouvez accepter ou rejeter un certificat qui vous Compte AWS a été transféré Compte AWS par un autre en utilisant la AWS IoT console ou le AWS CLI.

Accepter ou refuser un transfert de certificat (console)

Pour effectuer cette procédure, vous avez besoin de l'ID du certificat qui a été transféré sur votre compte.

Effectuez cette procédure depuis le compte recevant le certificat qui a été transféré.

Pour accepter ou refuser un certificat qui a été transféré à votre Compte AWS

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.

2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.

   Choisissez le certificat avec un statut transfert en attente que vous souhaitez accepter ou rejeter et ouvrez sa page détails.

3. Sur la page Détails du certificat, dans le menu Actions,

   - Pour accepter le certificat, choisissez Accepter le transfert.
   - Pour ne pas accepter le certificat, choisissez Rejeter le transfert.

Accepter ou refuser un transfert de certificat (CLI)

Pour terminer cette procédure, vous aurez besoin *certificateId* du transfert de certificat que vous souhaitez accepter ou refuser.

Effectuez cette procédure depuis le compte recevant le certificat qui a été transféré.

Pour accepter ou refuser un certificat qui a été transféré à votre Compte AWS

1. Pour accepter le certificat, utilisez la commande accept-certificate-transfer.

   ```
   aws iot accept-certificate-transfer --certificate-id certificateId
   ```

2. Pour refuser le certificat, utilisez la commande reject-certificate-transfer.

   ```
   aws iot reject-certificate-transfer --certificate-id certificateId
   ```

Annuler un transfert de certificat

Vous pouvez annuler un transfert de certificat avant qu'il ne soit accepté à l'aide de la AWS IoT console ou du AWS CLI.

Annuler un transfert de certificat (console)

Pour effectuer cette procédure, vous aurez besoin de l'ID du transfert de certificat que vous souhaitez annuler.

Effectuez cette procédure à partir du compte qui a initié le transfert du certificat.

Pour annuler un transfert de certificat

1. Connectez-vous à la console AWS de gestion et AWS IoT ouvrez-la.
2. Dans le volet de navigation de gauche, choisissez Sécurité, puis Certificats.

   Choisissez le certificat avec le statut Transféré dont vous souhaitez annuler le transfert et ouvrez son menu d'options.
3. Dans le menu des options du certificat, choisissez l'option Révoquer le transfert pour annuler le transfert du certificat.

   > **Important**
   >
   > Veuillez à ne pas confondre l'option Révoquer le transfert avec l'option Révoquer. L'option Révoquer le transfert annule le transfert du certificat, tandis que l'option Révoquer rend le certificat irréversiblement inutilisable par AWS IoT.

Annuler un transfert de certificat (CLI)

Pour terminer cette procédure, vous aurez besoin *certificateId* du transfert de certificat que vous souhaitez annuler.

Effectuez cette procédure à partir du compte qui a initié le transfert du certificat.

Utilisez la commande cancel-certificate-transfer pour annuler le transfert du certificat.

```
aws iot cancel-certificate-transfer --certificate-id certificateId
```

Validation personnalisée du certificat client

AWS IoT Core prend en charge la validation personnalisée des certificats client pour les certificats client X.509, ce qui améliore la gestion de l'authentification des clients. Cette méthode de validation des certificats est également connue sous le nom de vérifications de certificats de pré-authentification, dans lesquelles vous évaluez les certificats clients en fonction de vos propres critères (définis dans une fonction Lambda) et révoquez les certificats clients ou le certificat d'autorité de signature (CA) des certificats pour empêcher les clients de se connecter à. AWS IoT Core Par exemple, vous pouvez créer vos propres contrôles de révocation des certificats qui valident le statut des certificats par rapport aux autorités de validation qui prennent en charge les points de terminaison du protocole OCSP (Online Certificate Status Protocol) ou des listes de révocation de

certificats (CRL), et qui empêchent les clients dont les certificats ont été révoqués de se connecter. Les critères utilisés pour évaluer les certificats clients sont définis dans une fonction Lambda (également appelée Lambda de pré-authentification). Vous devez utiliser les points de terminaison définis dans les configurations de domaine et le type d'authentification doit être le certificat X.509. En outre, les clients doivent fournir l'extension SNI (Server Name Indication) lorsqu'ils se connectent à AWS IoT Core.

> **Note**
>
> Cette fonctionnalité n'est pas prise en charge dans les AWS GovCloud (US) régions.

Le processus de validation du certificat client personnalisé implique les étapes suivantes.

- Étape 1 : enregistrez vos certificats clients X.509 avec AWS IoT Core
- Étape 2 : création d'une fonction Lambda
- Étape 3 : Autoriser AWS IoT l'appel de votre fonction Lambda
- Étape 4 : définir la configuration de l'authentification pour un domaine

Étape 1 : enregistrez vos certificats clients X.509 avec AWS IoT Core

Si ce n'est pas déjà fait, enregistrez et activez vos certificats clients X.509 avec AWS IoT Core. Sinon, passez à l'étape suivante.

Pour enregistrer et activer vos certificats clients auprès de AWS IoT Core, suivez les étapes suivantes :

1. Si vous créez des certificats clients directement avec AWS IoT. Ces certificats clients seront automatiquement enregistrés auprès de AWS IoT Core.
2. Si vous créez vos propres certificats clients, suivez ces instructions pour les enregistrer auprès de AWS IoT Core.
3. Pour activer vos certificats clients, suivez ces instructions.

Étape 2 : création d'une fonction Lambda

Vous devez créer une fonction Lambda qui effectuera la vérification des certificats et sera appelée à chaque tentative de connexion client pour le point de terminaison configuré. Lorsque vous créez cette fonction Lambda, suivez les instructions générales de la section Créez votre première fonction

Lambda. En outre, assurez-vous que la fonction Lambda respecte les formats de demande et de réponse attendus, comme suit :

Exemple d'événement lié à une fonction Lambda

```
{
 "connectionMetadata": {
  "id": "string"
 },
 "principalId": "string",
 "serverName": "string",
 "clientCertificateChain": [
  "string",
  "string"
 ]
}
```

`connectionMetadata`

Métadonnées ou informations supplémentaires relatives à la connexion du client à AWS IoT Core.

`principalId`

Identifiant principal associé au client dans la connexion TLS.

`serverName`

Chaîne de nom d'hôte SNI (Server Name Indication). AWS IoT Core exige que les appareils envoient l'extension SNI au protocole TLS (Transport Layer Security) et fournissent l'adresse complète du point de terminaison sur le `host_name` terrain.

`clientCertificateChain`

Le tableau de chaînes qui représente la chaîne de certificats X.509 du client.

Exemple de réponse de la fonction Lambda

```
{
 "isAuthenticated": "boolean"
}
```

`isAuthenticated`

Valeur booléenne qui indique si la demande est authentifiée.

> **Note**
>
> Dans la réponse Lambda, `isAuthenticated` il faut procéder `true` à une authentification et à une autorisation supplémentaires. Sinon, le certificat client IoT peut être désactivé et l'authentification personnalisée avec les certificats client X.509 peut être bloquée pour une authentification et une autorisation supplémentaires.

Étape 3 : Autoriser AWS IoT l'appel de votre fonction Lambda

Après avoir créé la fonction Lambda, vous devez autoriser son appel AWS IoT à l'aide de la commande CLI add permission. Notez que cette fonction Lambda sera invoquée à chaque tentative de connexion à votre point de terminaison configuré. Pour plus d'informations, consultez Autoriser l'appel AWS IoT de votre fonction Lambda.

Étape 4 : définir la configuration de l'authentification pour un domaine

La section suivante décrit comment définir la configuration de l'authentification pour un domaine personnalisé à l'aide du AWS CLI.

Définir la configuration du certificat client pour un domaine (CLI)

Si vous n'avez pas de configuration de domaine, utilisez la commande create-domain-configurationCLI pour en créer une. Si vous avez déjà une configuration de domaine, utilisez la commande update-domain-configurationCLI pour mettre à jour la configuration du certificat client pour un domaine. Vous devez ajouter l'ARN de la fonction Lambda que vous avez créée à l'étape précédente.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
    --client-certificate-config 'clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type AWS_X509|CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT|HTTPS \
```

```
    --client-certificate-config '{"clientCertificateCallbackArn":"arn:aws:lambda:us-
east-2:123456789012:function:my-function:1"}'
```

`domain-configuration-name`

Nom de la configuration de domaine.

`authentication-type`

Type d'authentification de la configuration du domaine. Pour plus d'informations, consultez la section Choix d'un type d'authentification.

`application-protocol`

Protocole d'application avec lequel les appareils communiquent AWS IoT Core. Pour plus d'informations, consultez la section Choix d'un protocole d'application.

`client-certificate-config`

Objet qui spécifie la configuration de l'authentification client pour un domaine.

`clientCertificateCallbackArn`

Le nom de ressource Amazon (ARN) de la fonction Lambda qui AWS IoT invoque une couche TLS lors de l'établissement d'une nouvelle connexion. Pour personnaliser l'authentification client afin de valider un certificat client personnalisé, vous devez ajouter l'ARN de la fonction Lambda que vous avez créée à l'étape précédente.

Pour plus d'informations, consultez CreateDomainConfigurationet consultez le UpdateDomainConfigurationGuide de référence des AWS IoT API. Pour plus d'informations sur les configurations de domaine, consultez la section Configurations de domaine.

Utilisateurs, groupes et rôles IAM

Les utilisateurs, groupes et rôles IAM constituent les mécanismes standard de gestion de l'identité et de l'authentification dans AWS. Vous pouvez les utiliser pour vous connecter aux interfaces AWS IoT HTTP à l'aide du AWS SDK et AWS CLI.

Les rôles IAM permettent également d'accéder AWS IoT à d'autres AWS ressources de votre compte en votre nom. Par exemple, si vous souhaitez qu'un appareil publie son état dans une table DynamoDB, les rôles IAM AWS IoT permettent d'interagir avec Amazon DynamoDB. Pour en savoir plus, consultez Rôles IAM.

Pour les connexions au courtier de messages via HTTP, AWS IoT authentifie les utilisateurs, les groupes et les rôles à l'aide du processus de signature Signature version 4. Pour plus d'informations, consultez la section Signature des demandes d' AWS API.

Lorsqu'ils utilisent AWS Signature Version 4 avec AWS IoT, les clients doivent prendre en charge les éléments suivants dans leur implémentation du protocole TLS :

- TLS 1.2
- Validation de la signature de certificat RSA SHA-256
- Une des suites de chiffrement de la section Prise en charge des suites de chiffrement TLS

Pour plus d'informations, consultez Gestion des identités et des accès pour AWS IoT.

Identités Amazon Cognito

Amazon Cognito Identity vous permet de créer des informations d' AWS identification temporaires à privilèges limités à utiliser dans les applications mobiles et Web. Lorsque vous utilisez Amazon Cognito Identity, créez des groupes d'identités qui créent des identités uniques pour vos utilisateurs et authentifiez-les auprès de fournisseurs d'identité tels que Login with Amazon, Facebook et Google. Vous pouvez également utiliser les identités Amazon Cognito avec vos propres identités authentifiées par le développeur. Pour plus d'informations, consultez Identité Amazon Cognito.

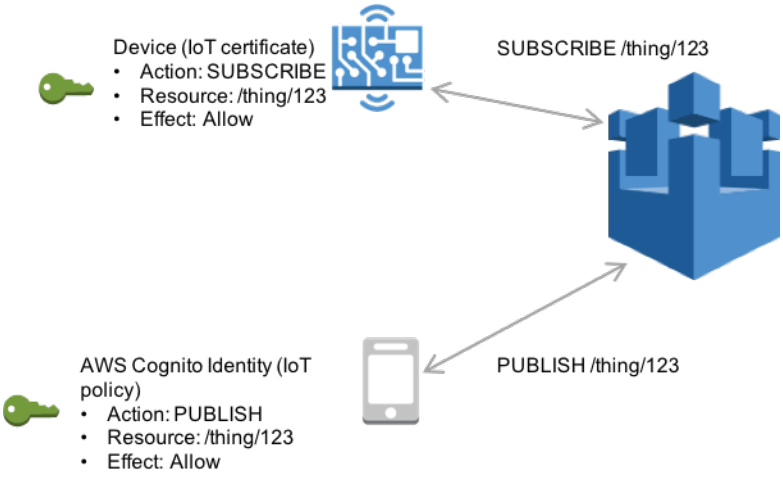
Pour utiliser Amazon Cognito Identity, définissez un pool d'identités Amazon Cognito associé à un rôle IAM. Le rôle IAM est associé à une politique IAM qui accorde aux identités de votre pool d'identités l'autorisation d'accéder à des AWS ressources telles que les services d'appel AWS .

L’identité Amazon Cognito crée des identités non authentifiées et authentifiées. Les identités non authentifiées sont utilisées pour les utilisateurs invités d'une application mobile ou Web qui souhaitent utiliser l'application sans se connecter. Les utilisateurs non authentifiés ne bénéficient que des autorisations spécifiées dans la politique IAM associée au réserve d'identités.

Lorsque vous utilisez des identités authentifiées, en plus de la politique IAM attachée au pool d'identités, vous devez associer une AWS IoT politique à une identité Amazon Cognito. Pour associer une AWS IoT politique, utilisez l' AttachPolicyAPI et accordez des autorisations à un utilisateur individuel de votre AWS IoT application. Vous pouvez utiliser cette AWS IoT politique pour attribuer des autorisations détaillées à des clients spécifiques et à leurs appareils.

Les utilisateurs authentifiés et non authentifiés sont des types d'identité différents. Si vous n'associez aucune AWS IoT politique à l'identité Amazon Cognito, un utilisateur authentifié ne parvient pas

à s'authentifier AWS IoT et n'a pas accès aux AWS IoT ressources et aux actions. Pour plus d'informations sur la création de politiques pour les identités Amazon Cognito, consultez Exemples de stratégie de publication/abonnement et Autorisation avec les identités Amazon Cognito.



Authentification et autorisation personnalisées

AWS IoT Core vous permet de définir des autorisateurs personnalisés afin que vous puissiez gérer vous-même l'authentification et l'autorisation de vos clients. Cela est utile lorsque vous devez utiliser des mécanismes d'authentification autres que ceux pris en charge de AWS IoT Core manière native. (Pour plus d'informations sur les mécanismes pris en charge de manière native, consultez the section called “Authentification client”).

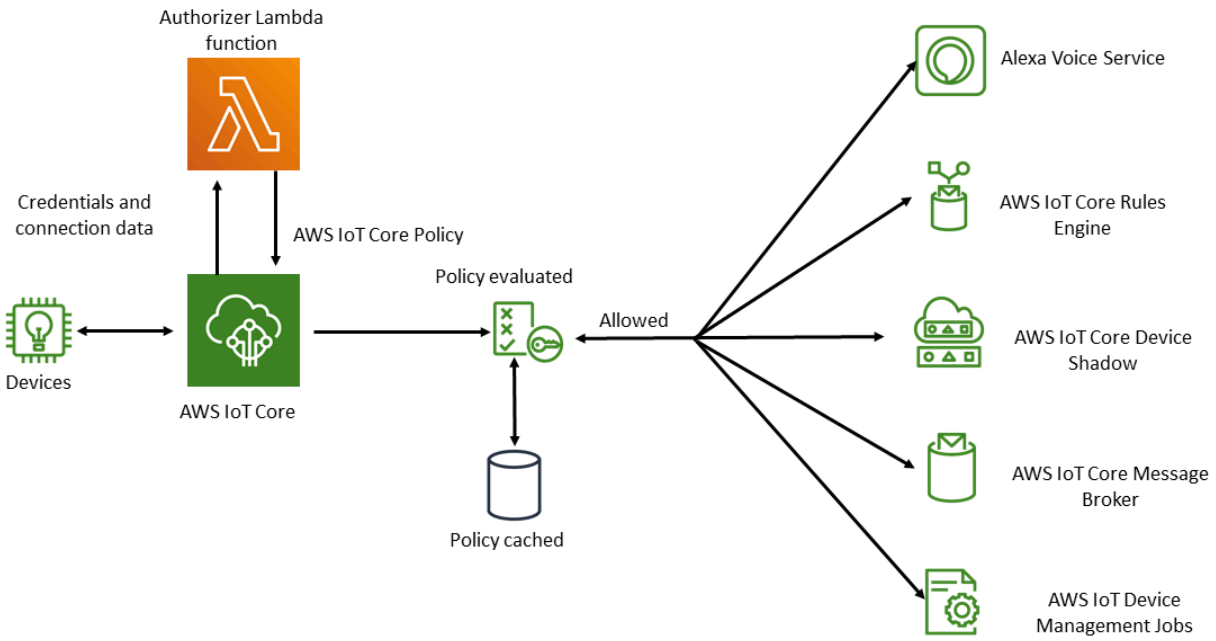
Par exemple, si vous migrez des appareils existants sur le terrain AWS IoT Core et que ces appareils utilisent un jeton porteur personnalisé ou un nom d'utilisateur et un mot de passe MQTT pour s'authentifier, vous pouvez les migrer vers ces appareils AWS IoT Core sans avoir à leur fournir de nouvelles identités. Vous pouvez utiliser l'authentification personnalisée avec tous les protocoles de communication pris en AWS IoT Core charge. Pour plus d'informations sur les protocoles pris en charge par AWS IoT Core , consultez the section called “Protocoles de communication des appareils”.

Rubriques

- Comprendre le flux de travail d'authentification personnalisé
- Création et gestion d'autorisateurs personnalisés (CLI)
- Authentification personnalisée avec des certificats clients X.509
- Connexion à l' AWS IoT Core aide de l'authentification personnalisée
- Dépannage de vos mécanismes d'autorisation

Comprendre le flux de travail d'authentification personnalisé

L'authentification personnalisée vous permet de définir comment authentifier et autoriser les clients à l'aide de ressources de mécanisme d'autorisation. Chaque autorisateur contient une référence à une fonction Lambda gérée par le client, une clé publique facultative pour valider les informations d'identification de l'appareil et des informations de configuration supplémentaires. Le schéma suivant illustre le flux de travail d'autorisation pour l'authentification personnalisée dans AWS IoT Core.



AWS IoT Core flux de travail d'authentification et d'autorisation personnalisé

La liste suivante explique chaque étape du flux de travail d'authentification et d'autorisation personnalisé.

1. Un appareil se connecte au point de terminaison de AWS IoT Core données d'un client à l'aide de l'une des solutions prises en chargethe section called “Protocoles de communication des appareils”. L'appareil transmet les informations d'identification soit dans les champs d'en-tête de la demande, soit dans les paramètres de requête (pour les WebSockets protocoles HTTP Publish ou MQTT over), soit dans le champ du nom d'utilisateur et du mot de passe du message MQTT CONNECT (pour les protocoles MQTT et MQTT over). WebSockets
2. AWS IoT Core vérifie l'une des deux conditions suivantes :

- La demande entrante spécifie un mécanisme d'autorisation.
- Un autorisateur par défaut est configuré pour le point de terminaison de AWS IoT Core données recevant la demande.

S'il AWS IoT Core trouve un autorisateur de l'une ou l'autre de ces manières, AWS IoT Core déclenche la fonction Lambda associée à l'autorisateur.

3. (Facultatif) Si vous avez activé la signature par jeton, AWS IoT Core valide la signature de la demande en utilisant la clé publique stockée dans l'autorisateur avant de déclencher la fonction Lambda. Si la validation échoue, AWS IoT Core arrête la demande sans invoquer la fonction Lambda.
4. La fonction Lambda reçoit les informations d'identification et les métadonnées de connexion dans la requête et prend une décision d'authentification.
5. La fonction Lambda renvoie les résultats de la décision d'authentification et un document de AWS IoT Core politique qui spécifie les actions autorisées dans la connexion. La fonction Lambda renvoie également des informations qui indiquent à quelle fréquence les informations d'identification contenues dans la demande sont AWS IoT Core revalidées en invoquant la fonction Lambda.
6. AWS IoT Core évalue l'activité de la connexion par rapport à la politique qu'elle a reçue de la fonction Lambda.
7. Une fois que la connexion est établie et que votre Lambda d'autorisation personnalisé est initialement invoqué, l'appel suivant peut être retardé jusqu'à 5 minutes sur les connexions inactives sans aucune opération MQTT. Ensuite, les appels suivants suivront l'intervalle d'actualisation dans votre autorisateur Lambda personnalisé. Cette approche permet d'éviter des appels excessifs susceptibles de dépasser la limite de simultanéité Lambda de votre compte. Compte AWS

Considérations relatives à la mise à l'échelle

Étant donné qu'une fonction Lambda gère l'authentification et l'autorisation de votre mécanisme d'autorisation, la fonction est soumise aux limites de tarification et de service Lambda, telles que le taux d'exécution simultanée. Pour plus d'informations sur la tarification Lambda, consultez Tarification Lambda. Vous pouvez gérer la charge sur votre fonction Lambda en ajustant les paramètres `refreshAfterInSeconds` et `disconnectAfterInSeconds` dans la réponse de votre fonction Lambda. Pour plus d'informations sur le contenu de la réponse de votre fonction Lambda, consultez the section called “Définir votre fonction Lambda”.

> **Note**
>
> Si vous laissez la signature activée, vous pouvez empêcher un déclenchement excessif de votre Lambda par des clients non reconnus. Considérez ceci avant de désactiver la connexion à votre mécanisme d'autorisation.

> **Note**
>
> Le délai d'expiration de la fonction Lambda pour le mécanisme d'autorisation personnalisé est de 5 secondes.

Création et gestion d'autorisateurs personnalisés (CLI)

AWS IoT Core implémente des schémas d'authentification et d'autorisation personnalisés en utilisant des autorisateurs personnalisés. Un autorisateur personnalisé est une AWS IoT Core ressource qui vous donne la flexibilité de définir et de mettre en œuvre les règles et les politiques en fonction de vos besoins spécifiques. Pour créer un autorisateur personnalisé avec des step-by-step instructions, voir Tutoriel : Création d'un autorisateur personnalisé pour. AWS IoT Core

Chaque mécanisme d'autorisation se compose des éléments suivants :

- Nom : Une chaîne unique définie par l'utilisateur qui identifie le mécanisme d'autorisation.
- ARN de la fonction Lambda : Amazon Resource Name (ARN) de la fonction Lambda qui implémente la logique d'autorisation et d'authentification.
- Nom de clé du jeton : nom de clé utilisé pour extraire le jeton des en-têtes HTTP, des paramètres de requête ou du nom d'utilisateur MQTT CONNECT afin d'effectuer la validation de la signature. Cette valeur est obligatoire si la signature est activée dans votre mécanisme d'autorisation.
- Indicateur de signature désactivée (facultatif) : valeur booléenne qui spécifie s'il faut désactiver l'exigence de signature sur les informations d'identification. Ceci est utile pour les scénarios dans lesquels la signature des informations d'identification n'a pas de sens, tels que les schémas d'authentification qui utilisent le nom d'utilisateur et le mot de passe MQTT. La valeur par défaut est `false`, la signature est donc activée par défaut.
- Clé publique de signature de jeton : clé publique que AWS IoT Core utilise pour valider la signature du jeton. Sa longueur minimale est de 2048 bits. Cette valeur est obligatoire si la signature est activée dans votre mécanisme d'autorisation.

Lambda vous facture le nombre d'exécutions de votre fonction Lambda et le temps nécessaire à l'exécution du code de votre fonction. Pour plus d'informations sur la tarification Lambda, consultez Tarification Lambda . Pour plus d'informations sur la création de fonctions Lambda, consultez le Guide du développeur Lambda.

> **Note**
>
> Si vous laissez la signature activée, vous pouvez empêcher un déclenchement excessif de votre Lambda par des clients non reconnus. Considérez ceci avant de désactiver la connexion à votre mécanisme d'autorisation.

> **Note**
>
> Le délai d'expiration de la fonction Lambda pour le mécanisme d'autorisation personnalisé est de 5 secondes.

Dans ce chapitre :

- Définir votre fonction Lambda
- Création d'un mécanisme d'autorisation
- Autorisation d' AWS IoT invoquer votre fonction Lambda
- Tester vos mécanismes d'autorisation
- Gestion des autorisations personnalisées

Définir votre fonction Lambda

Lorsque vous AWS IoT Core invoquez votre autorisateur, il déclenche le Lambda associé à l'autorisateur avec un événement contenant l'objet JSON suivant. L'exemple d'objet JSON contient tous les champs possibles. Tous les champs qui ne sont pas pertinents pour la demande de connexion ne sont pas inclus.

```
{
    "token" :"aToken",
    "signatureVerified": Boolean, // Indicates whether the device gateway has validated
 the signature.
```

```
    "protocols": ["tls", "http", "mqtt"], // Indicates which protocols to expect for
 the request.
    "protocolData": {
        "tls" : {
            "serverName": "serverName" // The server name indication (SNI) host_name
 string.
        },
        "http": {
            "headers": {
                "#{name}": "#{value}"
            },
            "queryString": "?#{name}=#{value}"
        },
        "mqtt": {
            "username": "myUserName",
            "password": "myPassword", // A base64-encoded string.
            "clientId": "myClientId" // Included in the event only when the device
 sends the value.
        }
    },
    "connectionMetadata": {
        "id": UUID // The connection ID. You can use this for logging.
    },
}
```

La fonction Lambda doit utiliser ces informations pour authentifier la connexion entrante et décider quelles actions sont autorisées dans la connexion. La fonction doit envoyer une réponse contenant les valeurs suivantes.

- `isAuthenticated`: Une valeur booléenne qui indique si la demande est authentifiée.
- `principalId`: Chaîne alphanumérique qui sert d'identifiant pour le jeton envoyé par la demande d'autorisation personnalisée. La valeur doit être une chaîne alphanumérique contenant au moins un et pas plus de 128 caractères et correspondre à ce modèle d'expression régulière (regex) : `([a-zA-Z0-9]){1,128}`. Les caractères spéciaux qui ne sont pas alphanumériques ne sont pas autorisés à être utilisés avec l'entrée`principalId`. AWS IoT Core Reportez-vous à la documentation des autres AWS services si des caractères spéciaux non alphanumériques sont autorisés pour le. `principalId`
- `policyDocuments`: liste des documents de AWS IoT Core politique au format JSON Pour plus d'informations sur la création de AWS IoT Core politiques, consultez. the section called “AWS

IoT Core politiques" Le nombre maximum de documents de stratégie est de 10 documents de stratégie. Chaque document de stratégie peut contenir un maximum de 2048 caractères.

- `disconnectAfterInSeconds` : Un entier qui spécifie la durée maximale (en secondes) de la connexion à la passerelle AWS IoT Core . La valeur minimale est de 300 secondes et la valeur maximale de 86 400 secondes. La valeur par défaut est 86 400.

> **Note**
>
> La valeur de `disconnectAfterInSeconds` (renvoyée par la fonction Lambda) est définie lorsque la connexion est établie. Cette valeur ne peut pas être modifiée lors des appels Lambda d'actualisation des politiques ultérieurs.

- `refreshAfterInSeconds`: Un entier qui qui spécifie l'intervalle entre les actualisations de stratégie. Lorsque cet intervalle est écoulé, AWS IoT Core appelle la fonction Lambda pour autoriser l'actualisation des stratégies. La valeur minimale est de 300 secondes et la valeur maximale de 86 400 secondes.

L'objet JSON suivant contient un exemple de réponse que votre fonction Lambda peut envoyer.

JSON

```
{
"isAuthenticated":true, //A Boolean that determines whether client can connect.
"principalId": "xxxxxxxx",  //A string that identifies the connection in logs.
"disconnectAfterInSeconds": 86400,
"refreshAfterInSeconds": 300,
 "policyDocuments": [
      {
        "Version": "2012-10-17",
        "Statement": [
           {
              "Action": "iot:Publish",
              "Effect": "Allow",
              "Resource": "arn:aws:iot:us-east-1:<your_aws_account_id>:topic/
customauthtesting"
            }
         ]
       }
    ]
```

```
}
```

La `policyDocument` valeur doit contenir un document AWS IoT Core de politique valide. Pour plus d'informations sur AWS IoT Core les politiques, consultezthe section called “AWS IoT Core politiques”. Dans MQTT sur TLS et MQTT sur WebSockets connexions, met en AWS IoT Core cache cette politique pendant l'intervalle spécifié dans la valeur du champ. `refreshAfterInSeconds` Dans le cas des connexions HTTP, la fonction Lambda est appelée pour chaque demande d'autorisation, sauf si votre appareil utilise des connexions HTTP persistantes (également appelées HTTP keep-alive ou HTTP connection reuse). Vous pouvez choisir d'activer la mise en cache lors de la configuration du mécanisme d'autorisation. Pendant cet intervalle, AWS IoT Core autorise les actions dans une connexion établie par rapport à cette politique mise en cache sans déclencher à nouveau votre fonction Lambda. En cas d'échec lors de l'authentification personnalisée, AWS IoT Core met fin à la connexion. AWS IoT Core met également fin à la connexion si elle est ouverte depuis plus longtemps que la valeur spécifiée dans le `disconnectAfterInSeconds` paramètre.

Vous trouverez ci-dessous JavaScript un exemple de fonction Lambda Node.js qui recherche un mot de passe dans le message MQTT Connect avec une valeur `test` égale à et renvoie une politique autorisant la connexion AWS IoT Core avec un client `myClientName` nommé et la publication sur une rubrique contenant le même nom de client. S'il ne trouve pas le mot de passe attendu, il renvoie une stratégie qui refuse ces deux actions.

```
// A simple Lambda function for an authorizer. It demonstrates
// how to parse an MQTT password and generate a response.

exports.handler = function(event, context, callback) {
    var uname = event.protocolData.mqtt.username;
    var pwd = event.protocolData.mqtt.password;
    var buff = new Buffer(pwd, 'base64');
    var passwd = buff.toString('ascii');
    switch (passwd) {
        case 'test':
            callback(null, generateAuthResponse(passwd, 'Allow'));
            break;
        default:
            callback(null, generateAuthResponse(passwd, 'Deny'));
    }
};

// Helper function to generate the authorization response.
var generateAuthResponse = function(token, effect) {
```

```
    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'TEST123';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var publishStatement = {};
    var connectStatement = {};
    connectStatement.Action = ["iot:Connect"];
    connectStatement.Effect = effect;
    connectStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:client/
myClientName"];
    publishStatement.Action = ["iot:Publish"];
    publishStatement.Effect = effect;
    publishStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:topic/telemetry/
myClientName"];
    policyDocument.Statement[0] = connectStatement;
    policyDocument.Statement[1] = publishStatement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 300;

    return authResponse;
}
```

La fonction Lambda précédente renvoie le JSON suivant lorsqu'elle reçoit le mot de passe `test` attendu dans le message MQTT Connect. Les valeurs des propriétés `password` et `principalId` seront les valeurs du message MQTT Connect.

```
{
  "password": "password",
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "iot:Connect",
          "Effect": "Allow",
          "Resource": "*"
        },
```

```
        {
          "Action": "iot:Publish",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        },
        {
          "Action": "iot:Subscribe",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topicfilter/telemetry/
${iot:ClientId}"
        },
        {
          "Action": "iot:Receive",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        }
      ]
    }
  ],
  "disconnectAfterInSeconds": 3600,
  "refreshAfterInSeconds": 300
}
```

Création d'un mécanisme d'autorisation

Vous pouvez créer un autorisateur à l'aide de l'CreateAuthorizerAPI. L'exemple suivant décrit la commande.

```
aws iot create-authorizer
--authorizer-name MyAuthorizer
--authorizer-function-arn arn:aws:lambda:us-
west-2:<account_id>:function:MyAuthorizerFunction  //The ARN of the Lambda function.
[--token-key-name MyAuthorizerToken //The key used to extract the token from headers.
[--token-signing-public-keys FirstKey=
 "-----BEGIN PUBLIC KEY-----
  [...insert your public key here...]
  -----END PUBLIC KEY-----"
[--status ACTIVE]
[--tags <value>]
[--signing-disabled | --no-signing-disabled]
```

Vous pouvez utiliser le paramètre `signing-disabled` pour désactiver la validation de signature pour chaque invocation de votre mécanisme d'autorisation. Nous vous recommandons fortement de

ne pas désactiver la signature, sauf si vous y êtes obligé. La validation de signature vous protège contre les appels excessifs de votre fonction Lambda à partir d'appareils inconnus. Vous ne pouvez pas mettre à jour l'état `signing-disabled` d'un organisme d'autorisation après l'avoir créé. Pour modifier ce comportement, vous devez créer un autre mécanisme d'autorisation personnalisé avec une valeur différente pour le paramètre `signing-disabled`.

Les valeurs des paramètres `tokenKeyName` et `tokenSigningPublicKeys` sont facultatives si vous avez désactivé la signature. Ce sont des valeurs obligatoires si la signature est activée.

Après avoir créé votre fonction Lambda et l'autorisateur personnalisé, vous devez explicitement accorder au AWS IoT Core service l'autorisation d'appeler la fonction en votre nom. Vous pouvez le faire à l'aide de la commande suivante.

> **Note**
>
> Le point de terminaison IoT par défaut peut ne pas prendre en charge l'utilisation d'autorisateurs personnalisés dotés de fonctions Lambda. Vous pouvez plutôt utiliser des configurations de domaine pour définir un nouveau point de terminaison, puis spécifier ce point de terminaison pour l'autorisateur personnalisé.

```
aws lambda add-permission --function-name <lambda_function_name>
--principal iot.amazonaws.com --source-arn <authorizer_arn>
--statement-id Id-123 --action "lambda:InvokeFunction"
```

Autorisation d' AWS IoT invoquer votre fonction Lambda

Dans cette section, vous allez autoriser la ressource d'autorisation personnalisée que vous venez de créer pour exécuter la fonction Lambda. Pour accorder l'autorisation, vous pouvez utiliser la commande CLI add-permission.

Accordez l'autorisation à votre fonction Lambda à l'aide du AWS CLI

1. Après avoir inséré vos valeurs, entrez la commande suivante. Notez que la valeur `statement-id` doit être unique. Remplacez *`Id-1234`* par la valeur exacte que vous avez, sinon vous risquez de recevoir une `ResourceConflictException` erreur.

   ```
   aws lambda add-permission  \
   --function-name "custom-auth-function" \
   --principal "iot.amazonaws.com" \
   ```

```
--action "lambda:InvokeFunction" \
--statement-id "Id-1234" \
--source-arn authorizerArn
```

2. Si la commande aboutit, elle renvoie une instruction d'autorisation, comme dans cet exemple. Vous pouvez passer à la section suivante pour tester le mécanisme d'autorisation personnalisée.

```
{
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal
\":{\"Service\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction
\",\"Resource\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
auth-function\",\"Condition\":{\"ArnLike\":{\"AWS:SourceArn\":
\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\"}}}"
}
```

Si la commande échoue, elle renvoie une erreur, comme dans cet exemple. Vous devez vérifier et corriger l'erreur avant de continuer.

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function
```

Tester vos mécanismes d'autorisation

Vous pouvez utiliser l'TestInvokeAuthorizerAPI pour tester les valeurs d'invocation et de retour de votre autorisateur. Cette API vous permet de spécifier les métadonnées du protocole et de tester la validation de signature dans votre autorisateur.

Les onglets suivants montrent comment utiliser le AWS CLI pour tester votre autorisateur.

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER \
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER ^
```

```
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER `
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

La valeur du paramètre `token-signature` est le jeton signé. Pour savoir comment obtenir cette valeur, consultez the section called “Signer le jeton”.

Si votre mécanisme d'autorisation prend un nom d'utilisateur et un mot de passe, vous pouvez transmettre ces informations en utilisant le paramètre `--mqtt-context`. Les onglets suivants montrent comment utiliser l'API `TestInvokeAuthorizer` pour envoyer un objet JSON contenant un nom d'utilisateur, un mot de passe et un nom de client à votre mécanisme d'autorisation personnalisé.

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  \
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  ^
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  `
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Le mot de passe doit être codé en base64. L'exemple suivant montre comment coder un mot de passe dans un environnement de type Unix.

```
echo -n PASSWORD | base64
```

Gestion des autorisations personnalisées

Vous pouvez gérer vos autorisateurs à l'aide des méthodes suivantes APIs.

- ListAuthorizers: Afficher tous les autorisateurs de votre compte.
- DescribeAuthorizer: affiche les propriétés de l'autorisateur spécifié. Ces valeurs incluent la date de création, la date de la dernière modification et d'autres attributs.
- SetDefaultAuthorizer: Spécifie l'autorisateur par défaut pour vos points de terminaison AWS IoT Core de données. AWS IoT Core utilise cet autorisateur si un appareil ne transmet pas les AWS IoT Core informations d'identification et ne spécifie pas d'autorisateur. Pour plus d'informations sur l'utilisation des AWS IoT Core informations d'identification, consultezthe section called “Authentification client”.
- UpdateAuthorizer: modifie le statut, le nom de la clé du jeton ou les clés publiques de l'autorisateur spécifié.
- DeleteAuthorizer: Supprime l'autorisateur spécifié.

> **Note**
>
> Vous ne pouvez pas mettre à jour les exigences de signature d'un mécanisme d'autorisation. Cela signifie que vous ne pouvez pas désactiver la connexion si un mécanisme d'autorisation existant l'exige. Vous ne pouvez pas non plus exiger la connexion d’un mécanisme d’autorisation existant qui ne l’exige pas.

Authentification personnalisée avec des certificats clients X.509

Lorsque vous connectez des appareils à AWS IoT Core, plusieurs types d'authentification sont disponibles. Vous pouvez utiliser des certificats client X.509 qui peuvent être utilisés pour authentifier les connexions des clients et des appareils, ou définir des autorisateurs personnalisés pour gérer votre propre logique d'authentification et d'autorisation des clients. Cette rubrique explique comment utiliser l'authentification personnalisée avec les certificats client X.509.

L'utilisation de l'authentification personnalisée avec des certificats X.509 peut être utile si vous avez déjà authentifié vos appareils à l'aide de certificats X.509 et que vous souhaitez effectuer une validation supplémentaire et une autorisation personnalisée. Par exemple, si vous stockez

les données de vos appareils, telles que leurs numéros de série, dans le certificat client X.509, après avoir AWS IoT Core authentifié le certificat client X.509, vous pouvez utiliser un autorisateur personnalisé pour identifier des appareils spécifiques en fonction des informations stockées dans le champ du certificat. CommonName L'utilisation de l'authentification personnalisée avec des certificats X.509 peut améliorer la gestion de la sécurité de vos appareils lors de la connexion des appareils AWS IoT Core et offre plus de flexibilité pour gérer la logique d'authentification et d'autorisation. AWS IoT Core prend en charge l'authentification personnalisée avec des certificats X.509 utilisant le certificat X.509 et le type d'authentification d'autorisation personnalisé, qui fonctionne à la fois avec le protocole MQTT et le protocole HTTPS. Pour plus d'informations sur les types d'authentification et les protocoles d'application pris en charge par les terminaux des AWS IoT Core appareils, consultez la section Protocoles de communication des appareils.

> **Note**
>
> L'authentification personnalisée à l'aide de certificats clients X.509 n'est pas prise en charge dans les AWS GovCloud (US) régions.

> **Important**
>
> Vous devez utiliser un point de terminaison créé à l'aide de configurations de domaine. En outre, les clients doivent fournir l'extension SNI (Server Name Indication) lorsqu'ils se connectent à AWS IoT Core.

Le processus d'authentification des appareils à l'aide d'une authentification personnalisée avec des certificats clients X.509 comprend les étapes suivantes.

- Étape 1 : enregistrez vos certificats clients X.509 avec AWS IoT Core
- Étape 2 : création d'une fonction Lambda
- Étape 3 : créer un autorisateur personnalisé
- Étape 4 : définir le type d'authentification et le protocole d'application dans une configuration de domaine

Étape 1 : enregistrez vos certificats clients X.509 avec AWS IoT Core

Si ce n'est pas déjà fait, enregistrez et activez vos certificats clients X.509 avec AWS IoT Core. Sinon, passez à l'étape suivante.

Pour enregistrer et activer vos certificats clients auprès de AWS IoT Core, suivez les étapes suivantes :

1. Si vous créez des certificats clients directement avec AWS IoT. Ces certificats clients seront automatiquement enregistrés auprès de AWS IoT Core.
2. Si vous créez vos propres certificats clients, suivez ces instructions pour les enregistrer auprès de AWS IoT Core.
3. Pour activer vos certificats clients, suivez ces instructions.

Étape 2 : création d'une fonction Lambda

AWS IoT Core utilise des autorisateurs personnalisés pour implémenter des schémas d'authentification et d'autorisation personnalisés. Un autorisateur personnalisé est associé à une fonction Lambda qui détermine si un appareil est authentifié et quelles opérations le périphérique est autorisé à effectuer. Lorsqu'un appareil se connecte à AWS IoT Core, AWS IoT Core récupère les détails de l'autorisateur, y compris le nom de l'autorisateur et la fonction Lambda associée, et invoque la fonction Lambda. La fonction Lambda reçoit un événement contenant un objet JSON contenant les données du certificat client X.509 du périphérique. Votre fonction Lambda utilise cet objet JSON d'événement pour évaluer la demande d'authentification, décider des actions à entreprendre et renvoyer une réponse.

Exemple d'événement lié à une fonction Lambda

L'exemple d'objet JSON suivant contient tous les champs possibles qui peuvent être inclus. L'objet JSON proprement dit ne contiendra que les champs relatifs à la demande de connexion spécifique.

```
{
 "token": "aToken",
 "signatureVerified": true,
 "protocols": [
  "tls",
  "mqtt"
 ],
 "protocolData": {
  "tls": {
   "serverName": "serverName",
   "x509CertificatePem": "x509CertificatePem",
   "principalId": "principalId"
  },
```

```
  "mqtt": {
   "clientId": "myClientId",
                    "username": "myUserName",
                    "password": "myPassword"
  }
 },
 "connectionMetadata": {
  "id": "UUID"
 }
}
```

`signatureVerified`

Valeur booléenne qui indique si la signature du jeton configurée dans l'autorisateur est vérifiée ou non avant d'appeler la fonction Lambda de l'autorisateur. Si l'autorisateur est configuré pour désactiver la signature par jeton, ce champ sera faux.

`protocols`

Un tableau qui contient les protocoles attendus pour la demande.

`protocolData`

Objet contenant des informations sur les protocoles utilisés dans la connexion. Il fournit des informations spécifiques au protocole qui peuvent être utiles pour l'authentification, l'autorisation, etc.

`tls`- Cet objet contient des informations relatives au protocole TLS (Transport Layer Security).

- `serverName`- La chaîne de nom d'hôte SNI (Server Name Indication). AWS IoT Core exige que les appareils envoient l'extension SNI au protocole TLS (Transport Layer Security) et fournissent l'adresse complète du point de terminaison sur le `host_name` terrain.
- `x509CertificatePem`- Le certificat X.509 au format PEM, qui est utilisé pour l'authentification du client dans la connexion TLS.
- `principalId`- L'identifiant principal associé au client dans la connexion TLS.

`mqtt`- Cet objet contient des informations relatives au protocole MQTT.

- `clientId`- Une chaîne ne doit être incluse que dans le cas où l'appareil envoie cette valeur.
- `username`- Le nom d'utilisateur fourni dans le paquet MQTT Connect.
- `password`- Le mot de passe fourni dans le paquet MQTT Connect.

`connectionMetadata`

Métadonnées de la connexion.

`id`- L'identifiant de connexion, que vous pouvez utiliser pour la journalisation et le dépannage.

> **Note**
>
> Dans ce cas, il s'agit d'un objet JSON `x509CertificatePem` et `principalId` deux nouveaux champs figurent dans la demande. La valeur de `principalId` est identique à la valeur de`certificateId`. Pour plus d'informations, consultez la section Certificat.

Exemple de réponse de la fonction Lambda

La fonction Lambda doit utiliser les informations de l'objet JSON d'événement pour authentifier la connexion entrante et décider quelles actions sont autorisées dans la connexion.

L'objet JSON suivant contient un exemple de réponse que votre fonction Lambda peut envoyer.

```
{
 "isAuthenticated": true,
 "principalId": "xxxxxxxx",
 "disconnectAfterInSeconds": 86400,
 "refreshAfterInSeconds": 300,
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
    {
     "Effect": "Allow",
     "Action": "iot:Publish",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/customauthtesting"
    }
   ]
  }
 ]
}
```

Dans cet exemple, cette fonction doit envoyer une réponse contenant les valeurs suivantes.

`isAuthenticated`

Valeur booléenne qui indique si la demande est authentifiée.

`principalId`

Chaîne alphanumérique qui sert d'identifiant pour le jeton envoyé par la demande d'autorisation personnalisée. La valeur doit être une chaîne alphanumérique comportant au moins un caractère, mais pas plus de 128. Il identifie la connexion dans les journaux. La valeur de `principalId` doit être identique à la valeur de l'objet JSON `principalId` de l'événement (c'est-à-dire le certificateID du certificat X.509).

`policyDocuments`

Liste des documents de politique au format JSON AWS IoT Core . La valeur est facultative et prend en charge les variables de politique d'objet et les variables de politique de certificat. Le nombre maximum de documents de politique est de 10. Chaque document de stratégie peut contenir un maximum de 2048 caractères. Si plusieurs politiques sont associées à votre certificat client et à la fonction Lambda, l'autorisation est un ensemble de toutes les politiques. Pour plus d'informations sur la création de AWS IoT Core politiques, consultez la section Politiques.

`disconnectAfterInSeconds`

Nombre entier qui indique la durée maximale (en secondes) de la connexion à la AWS IoT Core passerelle. La valeur minimale est de 300 secondes et la valeur maximale de 86 400 secondes. `disconnectAfterInSeconds`est valable pour la durée de vie d'une connexion et n'est pas actualisé lors de mises à jour consécutives des politiques.

`refreshAfterInSeconds`

Nombre entier qui indique l'intervalle entre les actualisations des politiques. Lorsque cet intervalle est dépassé, AWS IoT Core invoque la fonction Lambda pour permettre l'actualisation des politiques. La valeur minimale est de 300 secondes et la valeur maximale de 86 400 secondes.

Exemple de fonction Lambda

Voici un exemple de fonction Lambda de Node.js. La fonction examine le certificat X.509 du client et extrait les informations pertinentes telles que le numéro de série, l'empreinte digitale et le nom du sujet. Si les informations extraites correspondent aux valeurs attendues, le client est autorisé à se connecter. Ce mécanisme garantit que seuls les clients autorisés possédant des certificats valides peuvent établir une connexion.

```
const crypto = require('crypto');

exports.handler = async (event) => {

    // Extract the certificate PEM from the event
    const certPem = event.protocolData.tls.x509CertificatePem;

    // Parse the certificate using Node's crypto module
    const cert = new crypto.X509Certificate(certPem);

    var effect = "Deny";
    // Allow permissions only for a particular certificate serial, fingerprint, and
 subject
    if (cert.serialNumber === "7F8D2E4B9C1A5036DE8F7C4B2A91E5D80463BC9A1257" // This is
 a random serial
       && cert.fingerprint ===
 "F2:9A:C4:1D:B5:E7:08:3F:6B:D0:4E:92:A7:C1:5B:8D:16:0F:E3:7A" // This is a random
 fingerprint
       && cert.subject === "allow.example.com") {
      effect = "Allow";
    }

    return generateAuthResponse(event.protocolData.tls.principalId, effect);
};


// Helper function to generate the authorization response.
function generateAuthResponse(principalId, effect) {
    const authResponse = {
        isAuthenticated: true,
        principalId,
        disconnectAfterInSeconds: 3600,
        refreshAfterInSeconds: 300,
        policyDocuments: [
          {
            Version: "2012-10-17",
            Statement: [
              {
                Action: ["iot:Connect"],
                Effect: effect,
                Resource: [
                  "arn:aws:iot:us-east-1:123456789012:client/myClientName"
                ]
```

```
            },
            {
              Action: ["iot:Publish"],
              Effect: effect,
              Resource: [
                "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
              ]
            },
            {
              Action: ["iot:Subscribe"],
              Effect: effect,
              Resource: [
                 "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
              ]
            },
            {
              Action: ["iot:Receive"],
              Effect: effect,
              Resource: [
                 "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
              ]
            }
          ]
        }
      ]
    };

  return authResponse;
}
```

La fonction Lambda précédente renvoie le code JSON suivant lorsqu'elle reçoit un certificat avec le numéro de série, l'empreinte digitale et le sujet attendus. La valeur de `x509CertificatePem` sera le certificat client fourni lors de la poignée de contact TLS. Pour plus d'informations, consultez la section Définition de votre fonction Lambda.

```
{
 "isAuthenticated": true,
 "principalId": "principalId in the event JSON object",
 "policyDocuments": [
  {
   "Version": "2012-10-17",
   "Statement": [
```

```
    {
     "Action": "iot:Connect",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:client/myClientName"
    },
    {
     "Action": "iot:Publish",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    },
    {
     "Action": "iot:Subscribe",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/telemetry/
myClientName"
    },
    {
     "Action": "iot:Receive",
     "Effect": "Allow",
     "Resource": "arn:aws:iot:us-east-1:123456789012:topic/telemetry/myClientName"
    }
   ]
  }
 ],
 "disconnectAfterInSeconds": 3600,
 "refreshAfterInSeconds": 300
}
```

Étape 3 : créer un autorisateur personnalisé

Après avoir défini la fonction Lambda, créez un autorisateur personnalisé pour gérer votre propre logique d'authentification et d'autorisation du client. Vous pouvez suivre les instructions détaillées de l'étape 3 : créer une ressource d'autorisation client et son autorisation. Pour plus d'informations, consultez la section Création d'un autorisateur.

Lors de la création de l'autorisateur personnalisé, vous devez AWS IoT autoriser l'appel de la fonction Lambda après sa création. Pour obtenir des instructions détaillées, consultez Autoriser l'appel AWS IoT de votre fonction Lambda.

Étape 4 : définir le type d'authentification et le protocole d'application dans une configuration de domaine

Pour authentifier les appareils à l'aide d'une authentification personnalisée avec des certificats clients X.509, vous devez définir le type d'authentification et le protocole d'application dans une configuration de domaine, et vous devez envoyer l'extension SNI. La valeur de `authenticationType` doit êtreCUSTOM_AUTH_X509, et la valeur de `applicationProtocol` peut être SECURE_MQTT ouHTTPS.

Définissez le type d'authentification et le protocole d'application dans la configuration du domaine (CLI)

Si vous n'avez pas de configuration de domaine, utilisez la create-domain-configurationcommande pour en créer une. La valeur de `authenticationType` doit êtreCUSTOM_AUTH_X509, et la valeur de `applicationProtocol` peut être SECURE_MQTT ouHTTPS.

```
aws iot create-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

Si vous avez déjà une configuration de domaine, utilisez la update-domain-configurationcommande update `authenticationType` et `applicationProtocol` si nécessaire. Notez que vous ne pouvez pas modifier le type ou le protocole d'authentification sur le point de terminaison par défaut (`iot:Data-ATS`).

```
aws iot update-domain-configuration \
    --domain-configuration-name domainConfigurationName \
    --authentication-type CUSTOM_AUTH_X509 \
    --application-protocol SECURE_MQTT \
    --authorizer-config '{
        "defaultAuthorizerName": my-custom-authorizer
    }'
```

`domain-configuration-name`

Nom de la configuration de domaine.

`authentication-type`

Type d'authentification de la configuration du domaine. Pour plus d'informations, consultez la section Choix d'un type d'authentification.

`application-protocol`

Protocole d'application avec lequel les appareils communiquent AWS IoT Core. Pour plus d'informations, consultez la section Choix d'un protocole d'application.

`--authorizer-config`

Objet qui spécifie la configuration de l'autorisateur dans une configuration de domaine.

`defaultAuthorizerName`

Nom de l'autorisateur pour une configuration de domaine.

Pour plus d'informations, consultez CreateDomainConfigurationet consultez le UpdateDomainConfigurationGuide de référence des AWS IoT API. Pour plus d'informations sur la configuration des domaines, consultez la section Configurations des domaines.

Connexion à l' AWS IoT Core aide de l'authentification personnalisée

Les appareils peuvent se connecter à AWS IoT Core l'aide d'une authentification personnalisée avec n'importe quel protocole prenant AWS IoT Core en charge la messagerie des appareils. Pour plus d'informations sur les protocoles de communication pris en charge, consultez the section called “Protocoles de communication des appareils”. Les données de connexion que vous transmettez à votre fonction Lambda d'autorisation dépendent du protocole que vous utilisez. Pour plus d'informations sur la création de votre fonction Lambda d'autorisation, consultez the section called “Définir votre fonction Lambda”. Les sections suivantes expliquent comment se connecter pour s'authentifier à l'aide de chaque protocole pris en charge.

HTTPS

Les appareils qui envoient des données à AWS IoT Core l'aide de l'API HTTP Publish peuvent transmettre des informations d'identification par le biais d'en-têtes de requête ou de paramètres de requête dans leurs requêtes HTTP POST. Les appareils peuvent spécifier un mécanisme d'autorisation à invoquer à l'aide du paramètre d'en-tête `x-amz-customauthorizer-name` ou de requête. Si la signature de jetons est activée dans votre mécanisme d'autorisation, vous devez transmettre le *`token-key-name`* et `x-amz-customauthorizer-signature` dans les en-têtes

ou paramètres de requête. Notez que la *token-signature* valeur doit être codée en URL lorsque vous l'utilisez JavaScript depuis le navigateur.

> **Note**
>
> Le mécanisme d'autorisation client pour le protocole HTTPS prend uniquement en charge les opérations de publication. Pour plus d'informations sur le protocole HTTPS, consultez the section called “Protocoles de communication des appareils”.

Les exemples de requêtes suivants montrent comment transmettre ces paramètres dans les en-têtes et paramètres de requête.

```
//Passing credentials via headers
POST /topics/topic?qos=qos HTTP/1.1
Host: your-endpoint
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
x-amz-customauthorizer-name: authorizer-name

//Passing credentials via query parameters
POST /topics/topic?qos=qos&x-amz-customauthorizer-signature=token-signature&token-key-name=token-value HTTP/1.1
```

MQTT

Les appareils qui se connectent à AWS IoT Core l'aide d'une connexion MQTT peuvent transmettre des informations d'identification via les `password` champs `username` et des messages MQTT. La valeur `username` peut éventuellement contenir une chaîne de requête qui transmet des valeurs supplémentaires (y compris un jeton, une signature et un nom du mécanisme d'autorisation) à votre mécanisme d'autorisation. Vous pouvez utiliser cette chaîne de requête si vous souhaitez utiliser un schéma d'authentification basé sur un jeton au lieu des valeurs `password` et `username`.

> **Note**
>
> Les données du champ du mot de passe sont codées en base64 par. AWS IoT Core Votre fonction Lambda doit le décoder.

L'exemple suivant contient une chaîne `username` contenant des paramètres supplémentaires spécifiant un jeton et une signature.

```
username?x-amz-customauthorizer-name=authorizer-name&x-amz-customauthorizer-
signature=token-signature&token-key-name=token-value
```

Pour appeler un autorisateur, les appareils qui se connectent à l'aide AWS IoT Core de MQTT et d'une authentification personnalisée doivent se connecter sur le port 443. Ils doivent également transmettre l'extension TLS ALPN (Application Layer Protocol Negotiation) avec une valeur égale à `mqtt` et l'extension SNI (Server Name Indication) avec le nom d'hôte de leur AWS IoT Core point de terminaison de données. Pour éviter des erreurs potentielles, la valeur de `x-amz-customauthorizer-signature` doit être codée en URL. Nous recommandons fortement que les valeurs de `x-amz-customauthorizer-name` et `token-key-name` soient codées en URL. Pour plus d'informations sur ces valeurs, consultez the section called “Protocoles de communication des appareils”. Le AWS IoT Appareil SDKs, appareil mobile SDKs et client AWS IoT d'appareil V2 peut configurer ces deux extensions.

MQTT terminé WebSockets

Les appareils qui se connectent à AWS IoT Core l'aide de MQTT over WebSockets peuvent transmettre des informations d'identification de l'une des deux manières suivantes.

- Via les en-têtes de requête ou les paramètres de requête contenus dans la demande HTTP UPGRADE pour établir la WebSockets connexion.
- Via les champs `username` et `password` dans le message MQTT CONNECT.

Si vous transmettez les informations d'identification via le message de connexion MQTT, les extensions ALPN et SNI TLS sont requises. Pour plus d'informations sur ces extensions, consultez the section called “MQTT”. L'exemple suivant montre comment transmettre les informations d'identification via la demande de mise à niveau HTTP.

```
GET /mqtt HTTP/1.1
Host: your-endpoint
Upgrade: WebSocket
Connection: Upgrade
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
sec-WebSocket-Key: any random base64 value
```

```
sec-websocket-protocol: mqtt
sec-WebSocket-Version: websocket version
```

Signer le jeton

Vous devez signer le jeton avec la clé privée de la paire de clés publique-privée que vous avez utilisée lors de l'appel `create-authorizer`. Les exemples suivants montrent comment créer la signature du jeton à l'aide d'une commande de type Unix et. JavaScript Ils utilisent l'algorithme de hachage SHA-256 pour coder la signature.

Command line

```
echo -n TOKEN_VALUE | openssl dgst -sha256 -sign PEM encoded RSA private key |
 openssl base64
```

JavaScript

```
const crypto = require('crypto')

const key = "PEM encoded RSA private key"

const k = crypto.createPrivateKey(key)
let sign = crypto.createSign('SHA256')
sign.write(t)
sign.end()
const s = sign.sign(k, 'base64')
```

Dépannage de vos mécanismes d'autorisation

Cette rubrique passe en revue les problèmes courants des flux de travail d'authentification personnalisés et les étapes pour les résoudre. Pour résoudre les problèmes le plus efficacement possible, activez les CloudWatch journaux pour DEBUG AWS IoT Core et définissez le niveau de journalisation sur DEBUG. Vous pouvez activer CloudWatch les journaux dans la AWS IoT Core console (https://console.aws.amazon.com/iot/). Pour plus d'informations sur l'activation et la configuration des journaux pour AWS IoT Core, consultez the section called “Configuration de la AWS IoT journalisation”.

> **Note**
>
> Si vous laissez le niveau de journalisation au niveau DEBUG pendant de longues périodes, vous CloudWatch risquez de stocker de grandes quantités de données de journalisation. Cela peut augmenter vos CloudWatch frais. Envisagez d'utiliser la journalisation basée sur les ressources pour augmenter la verbosité uniquement pour les appareils d'un groupe d'objets particulier. Pour plus d'informations sur la journalisation basée sur les ressources, consultez the section called “Configuration de la AWS IoT journalisation”. De plus, une fois le dépannage terminé, réduisez le niveau du journal à un niveau moins détaillé.

Avant de commencer le dépannage, consultez the section called “Comprendre le flux de travail d'authentification personnalisé” pour un aperçu général du processus d'authentification personnalisé. Cela vous aide à comprendre où chercher la source d’un problème.

Cette rubrique aborde les deux domaines suivants que vous devez étudier.

- Problèmes liés à la fonction Lambda de votre mécanisme d'autorisation.
- Problèmes liés à votre appareil.

Recherchez les problèmes dans la fonction Lambda de votre mécanisme d'autorisation

Effectuez les étapes suivantes pour vous assurer que les tentatives de connexion de vos appareils invoquent votre fonction Lambda.

1. Vérifiez quelle fonction Lambda est associée à votre mécanisme d'autorisation.

   Vous pouvez le faire en appelant l'DescribeAuthorizerAPI ou en cliquant sur l'autorisateur souhaité dans la section Secure de la AWS IoT Core console.
2. Vérifiez les métriques d’invocation pour la fonction Lambda. Effectuez les étapes suivantes pour ce faire.

   a. Ouvrez la AWS Lambda console (https://console.aws.amazon.com/lambda/) et sélectionnez la fonction associée à votre autorisateur.

   b. Choisissez l’onglet Surveiller et affichez les métriques pour la période pertinente à votre problème.

3. Si vous ne voyez aucun appel, vérifiez qu'il AWS IoT Core est autorisé à appeler votre fonction Lambda. Si vous voyez des invocations, passez à l’étape suivante. Effectuez les étapes suivantes pour vérifier que votre fonction Lambda dispose des autorisations requises.

   a. Choisissez l'onglet Autorisations correspondant à votre fonction dans la AWS Lambda console.

   b. Recherchez la section Stratégie basée sur les ressources au bas de la page. Si votre fonction Lambda dispose des autorisations requises, la stratégie ressemble à l'exemple suivant.

      JSON

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "Id123",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-
east-1:111111111111:function:FunctionName",
      "Condition": {
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:iot:us-
east-1:111111111111:authorizer/AuthorizerName"
        },
        "StringEquals": {
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

   c. Cette politique accorde l'`InvokeFunction`autorisation d'accéder à votre fonction au AWS IoT Core directeur. Si vous ne le voyez pas, vous devrez l'ajouter à l'aide de

l'AddPermissionAPI. L'exemple suivant vous montre comment procéder en utilisant le AWS CLI.

```
aws lambda add-permission --function-name FunctionName --principal
 iot.amazonaws.com --source-arn AuthorizerARn --statement-id Id-123 --action
 "lambda:InvokeFunction"
```

4. Si vous voyez des invocations, vérifiez qu'il n'y a pas d'erreurs. Une erreur peut indiquer que la fonction Lambda ne gère pas correctement l'événement de connexion qui AWS IoT Core lui est envoyé.

   Pour plus d'informations sur la gestion de l'événement dans votre fonction Lambda, consultez the section called "Définir votre fonction Lambda". Vous pouvez utiliser la fonction de test de la AWS Lambda console (https://console.aws.amazon.com/lambda/) pour coder en dur les valeurs de test de la fonction afin de vous assurer que celle-ci gère correctement les événements.

5. Si vous voyez des invocations sans erreur, mais que vos appareils ne parviennent pas à se connecter (ou à publier, s'abonner et recevoir des messages), le problème peut être que la stratégie renvoyée par votre fonction Lambda n'accorde pas d'autorisations pour les actions que vos appareils effectuent. Effectuez les étapes suivantes pour déterminer si quelque chose ne va pas avec la stratégie renvoyée par la fonction.

   a. Utilisez une requête Amazon CloudWatch Logs Insights pour analyser les journaux sur une courte période afin de détecter les défaillances. L'exemple de requête suivant trie les événements par horodatage et recherche les échecs.

      ```
      display clientId, eventType, status, @timestamp | sort @timestamp desc | filter
       status = "Failure"
      ```

   b. Mettez à jour votre fonction Lambda pour enregistrer les données auxquelles elle renvoie AWS IoT Core et l'événement qui déclenche la fonction. Vous pouvez utiliser ces journaux pour inspecter la stratégie créée par la fonction.

6. Si vous voyez des invocations sans erreur, mais que vos appareils ne parviennent pas à se connecter (ou à publier, s'abonner et recevoir des messages), une autre raison peut être que votre fonction Lambda a dépassé le délai d'attente. Le délai d'expiration de la fonction Lambda pour le mécanisme d'autorisation personnalisé est de 5 secondes. Vous pouvez vérifier la durée de la fonction dans CloudWatch les journaux ou les métriques.

Enquête sur les problèmes liés aux appareils

Si vous ne rencontrez aucun problème lors de l'invocation de votre fonction Lambda ou avec la stratégie renvoyée par la fonction, recherchez les problèmes liés aux tentatives de connexion de vos appareils. Les demandes de connexion mal formées peuvent empêcher AWS IoT Core le déclenchement de votre autorisateur. Des problèmes de connexion peuvent survenir à la fois au niveau des couches TLS et application.

Problèmes possibles avec la couche TLS :

- Les clients doivent transmettre soit un en-tête de nom d'hôte (HTTP, MQTT par-dessus WebSockets), soit l'extension TLS avec indication du nom du serveur (HTTP, MQTT over WebSockets, MQTT) dans toutes les demandes d'authentification personnalisées. Dans les deux cas, la valeur transmise doit correspondre à l'un des points de terminaison de AWS IoT Core données de votre compte. Il s'agit des points de terminaison renvoyés lorsque vous exécutez les commandes CLI suivantes.
  - `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
  - `aws iot describe-endpoint --endpoint-type iot:Data`(pour les VeriSign terminaux existants)
- Les appareils qui utilisent l'authentification personnalisée pour les connexions MQTT doivent également transmettre l'extension TLS Application Layer Protocol Négociation (ALPN) avec une valeur de `mqtt`.
- L'authentification personnalisée est actuellement disponible uniquement sur le port 443.

Problèmes possibles au niveau de la couche application :

- Si la signature est activée (le champ `signingDisabled` est faux dans votre mécanisme d'autorisation), recherchez les problèmes de signature suivants.
  - Assurez-vous de transmettre la signature du jeton dans l'en-tête `x-amz-customauthorizer-signature` ou dans un paramètre de chaîne de requête.
  - Assurez-vous que le service ne signe pas une valeur autre que le jeton.
  - Assurez-vous de transmettre le jeton dans le paramètre d'en-tête ou de requête que vous avez spécifié dans le champ `token-key-name` de votre mécanisme d'autorisation.
- Assurez-vous que le nom du mécanisme d'autorisation que vous transmettez dans le paramètre d'en-tête `x-amz-customauthorizer-name` ou de chaîne de requête est valide ou qu'un mécanisme d'autorisation par défaut est défini pour votre compte.

Autorisation

L'autorisation est le processus d'octroi d'autorisations à une identité authentifiée. Vous accordez des autorisations lors de AWS IoT Core l'utilisation AWS IoT Core des politiques IAM. Cette rubrique couvre les stratégies AWS IoT Core . Pour plus d'informations sur les politiques IAM, consultez Gestion des identités et des accès pour AWS IoT et Comment AWS IoT fonctionne avec IAM.

AWS IoT Core les politiques déterminent ce que peut faire une identité authentifiée. Une identité authentifiée peut être utilisée par des appareils, des applications mobiles, des applications web et des applications de bureau. Une identité authentifiée peut même être un utilisateur qui saisit des commandes AWS IoT Core CLI. Une identité ne peut exécuter AWS IoT Core des opérations que si elle dispose d'une politique qui lui accorde l'autorisation de ces opérations.

Les AWS IoT Core politiques et les politiques IAM sont utilisées AWS IoT Core pour contrôler les opérations qu'une identité (également appelée principal) peut effectuer. Le type de politique que vous utilisez dépend du type d'identité que vous utilisez pour vous authentifier. AWS IoT Core

AWS IoT Core les opérations sont divisées en deux groupes :

- L'API de plan de contrôle vous permet d'effectuer des tâches administratives telles que la création ou la mise à jour de certificats, d'objets, de règles, etc.
- L'API du plan de données vous permet d'envoyer des données vers et de recevoir des données depuis AWS IoT Core.

Le type de stratégie que vous utilisez varie selon que vous utilisez l'API de plan de contrôle ou l'API de plan de données.

Le tableau suivant présente les différents types d'identité, les protocoles qu'ils utilisent, ainsi que les types de stratégie qui peuvent être utilisés à des fins d'autorisation.

AWS IoT Core API de plan de données et types de politiques

Protocole et mécanisme d'authentification	SDK	Type d'identité	Type de stratégie		
MQTT sur TLS/TCP, authentification mutuelle TLS (port 8883 ou 443)[‡])	AWS IoT SDK de l'appareil	Certificats X.509	AWS IoT Core politique		
MQTT sur HTTPS/WebSocket, authentification AWS SigV4 (port 443)	AWS SDK mobile	Identité Amazon Cognito authentifiée	IAM et politiques AWS IoT Core		
		Identité Amazon Cognito non authentifiée	Politique IAM		
		IAM, ou identité fédérée	Politique IAM		
HTTPS, authentification par AWS signature version 4 (port 443)	AWS CLI	Amazon Cognito, IAM ou identité fédérée	Politique IAM		

Protocole et mécanisme d'authentification	SDK	Type d'identité	Type de stratégie		
HTTPS, authentification mutuelle TLS (port 8443)	Pas de prise en charge SDK	Certificats X.509	AWS IoT Core politique		
HTTPS sur authentification personnalisée (Port 443)	AWS IoT SDK de l'appareil	Mécanisme d'autorisation personnalisé	Stratégie d'autorisation personnalisée		

AWS IoT Core API et types de politiques du plan de contrôle

Protocole et mécanisme d'authentification	SDK	Type d'identité	Type de stratégie		
Authentification par AWS signature HTTPS version 4 (port 443)	AWS CLI	Identité Amazon Cognito	Politique IAM		
		IAM, ou identité fédérée	Politique IAM		

AWS IoT Core les politiques sont associées aux certificats X.509, aux identités Amazon Cognito ou aux groupes d'objets. Les politiques IAM sont attachées à un utilisateur, un groupe ou un rôle IAM. Si vous utilisez la AWS IoT console ou la AWS IoT Core CLI pour associer la politique (à un certificat,

à Amazon Cognito Identity ou à un groupe d'objets), vous utilisez une AWS IoT Core stratégie. Dans le cas contraire, vous utilisez une politique IAM. AWS IoT Core les politiques associées à un groupe d'objets s'appliquent à tout élément de ce groupe d'objets. Pour que la AWS IoT Core politique entre en vigueur, le nom `clientId` et le nom de l'objet doivent correspondre.

L'autorisation basée sur la stratégie est un outil puissant. Elle vous permet de contrôler entièrement ce qu'un appareil, un utilisateur ou une application peut faire dans AWS IoT Core. Prenons l'exemple d'un appareil qui se connecte à l' AWS IoT Core aide d'un certificat. Vous pouvez autoriser l'appareil à accéder à toutes les rubriques MQTT ou limiter son accès à une seule rubrique. Autre exemple : imaginons le cas d'un utilisateur qui tape des commandes CLI dans la ligne de commande. En utilisant une politique, vous pouvez autoriser ou refuser l'accès à n'importe quelle commande ou AWS IoT Core ressource pour l'utilisateur. Vous pouvez également contrôler l'accès d'une application aux ressources AWS IoT Core .

Les modifications apportées à une stratégie peuvent prendre quelques minutes pour être effectives en raison de la façon dont AWS IoT met en cache les documents de stratégie. Autrement dit, l'accès à une ressource à laquelle l'accès a récemment été accordé peut prendre quelques minutes, et une ressource peut être accessible pendant plusieurs minutes après la révocation de son accès.

AWS formation et certification

Pour plus d'informations sur l'autorisation AWS IoT Core, suivez le cours Deep Dive into AWS IoT Core Authentication and Authorization sur le site Web de AWS formation et de certification.

AWS IoT Core politiques

AWS IoT Core les politiques sont des documents JSON. Elles suivent les mêmes conventions que les politiques IAM. AWS IoT Core prend en charge les politiques nommées afin que de nombreuses identités puissent faire référence au même document de politique. Les stratégies nommées comptent plusieurs versions afin de faciliter leur restauration.

AWS IoT Core les politiques vous permettent de contrôler l'accès au plan de AWS IoT Core données. Le plan de données AWS IoT Core comprend des opérations qui vous permettent de vous connecter au courtier de messages AWS IoT Core , d'envoyer et de recevoir des messages MQTT et d'obtenir ou de mettre à jour le Device Shadow d'un objet.

Une AWS IoT Core politique est un document JSON qui contient une ou plusieurs déclarations de politique. Chaque déclaration contient :

- `Effect`, qui indique si l'action est autorisée ou refusée.

- `Action`, qui indique l'action autorisée ou refusée par la stratégie.
- `Resource`, qui spécifie la ou les ressources sur lesquelles l'action est autorisée ou refusée.

Les modifications apportées à une politique peuvent prendre entre 6 et 8 minutes pour entrer en vigueur en raison de la façon dont les documents de politique sont mis en AWS IoT cache. Autrement dit, l'accès à une ressource à laquelle l'accès a récemment été accordé peut prendre quelques minutes, et une ressource peut être accessible pendant plusieurs minutes après la révocation de son accès.

AWS IoT Core des politiques peuvent être associées aux certificats X.509, aux identités Amazon Cognito et aux groupes d'objets. Les politiques attachées à un groupe d'objets s'appliquent à tout élément de ce groupe. Pour que la stratégie prenne effet, le `clientId` et le nom de l'objet doivent correspondre. les stratégies AWS IoT Core suivent la même logique d'évaluation que les politiques IAM. Par défaut, toutes les politiques sont implicitement refusées. Une autorisation explicite dans toute stratégie basée sur l'identité ou sur les ressources remplace le comportement par défaut. Un refus explicite dans n'importe quelle stratégie remplace toutes les autorisations. Pour plus d'informations, consultez Logique d'évaluation de la stratégie dans le AWS Identity and Access Management Guide d'utilisateur.

Rubriques

- AWS IoT Core actions politiques
- AWS IoT Core ressources d'action
- AWS IoT Core variables de politique
- Prévention du cas de figure de l'adjoint désorienté entre services
- AWS IoT Core exemples de politiques
- Autorisation avec les identités Amazon Cognito

AWS IoT Core actions politiques

Les actions de stratégie suivantes sont définies par AWS IoT Core :

Actions de stratégie MQTT

`iot:Connect`

Représente l'autorisation de se connecter au courtier de AWS IoT Core messages. L'autorisation `iot:Connect` est vérifiée chaque fois qu'une demande `CONNECT` est envoyée à l'agent. L'agent

de messages n'autorise pas deux clients avec le même ID client à rester connectés en même temps. Une fois que le deuxième client se connecte, l'agent ferme la connexion existante. Utilisez l'autorisation `iot:Connect` pour garantir que seuls les clients autorisés utilisant un ID client spécifique peuvent se connecter.

`iot:GetRetainedMessage`

Représente l'autorisation d'obtenir le contenu d'un seul message conservé. Les messages conservés sont les messages publiés avec l'indicateur RETAIN défini et stockés par AWS IoT Core. Pour obtenir l'autorisation d'obtenir une liste de tous les messages conservés du compte, consultez `iot:ListRetainedMessages`.

`iot:ListRetainedMessages`

Représente l'autorisation de récupérer des informations récapitulatives sur les messages conservés du compte, mais pas le contenu des messages. Les messages conservés sont les messages publiés avec l'indicateur RETAIN défini et stockés par AWS IoT Core. L'ARN de ressource spécifié pour cette action doit être `*`. Pour obtenir l'autorisation d'obtenir le contenu d'un seul message conservé, consultez `iot:GetRetainedMessage`.

`iot:Publish`

Représente l'autorisation de publier un sujet MQTT. Cette autorisation est vérifiée chaque fois qu'une demande PUBLISH est envoyée à l'agent. Vous pouvez l'utiliser pour permettre aux clients de publier sur des modèles de sujet spécifiques.

> **Note**
>
> Pour accorder l'autorisation `iot:Publish`, vous devez également accorder l'autorisation `iot:Connect`.

`iot:Receive`

Représente l'autorisation de recevoir un message de AWS IoT Core. L'autorisation `iot:Receive` est confirmée chaque fois qu'un message est remis à un client. Cette autorisation étant vérifiée à chaque diffusion, vous pouvez l'utiliser pour révoquer les autorisations des clients actuellement abonnés à une rubrique.

`iot:RetainPublish`

Représente l'autorisation de publier un message MQTT avec l'indicateur CONSERVER défini.

> **Note**
>
> Pour accorder l'autorisation `iot:RetainPublish`, vous devez également accorder l'autorisation `iot:Publish`.

`iot:Subscribe`

Représente l'autorisation de s'abonner à un filtre de rubrique. Cette autorisation est vérifiée chaque fois qu'une demande SUBSCRIBE est envoyée à l'agent. Utilisez-le pour permettre aux clients de s'abonner à des rubriques qui correspondent à des modèles spécifiques.

> **Note**
>
> Pour accorder l'autorisation `iot:Subscribe`, vous devez également accorder l'autorisation `iot:Connect`.

Actions de la stratégie Device Shadow

`iot:DeleteThingShadow`

Représente l'autorisation de supprimer le Device Shadow d'un objet. L'autorisation `iot:DeleteThingShadow` est vérifiée chaque fois qu'une demande est faite pour supprimer le contenu Device Shadow d'un objet.

`iot:GetThingShadow`

Représente l'autorisation de récupérer le Device Shadow d'un objet. L'autorisation `iot:GetThingShadow` est vérifiée chaque fois qu'une demande est faite pour récupérer le contenu Device Shadow d'un objet.

`iot:ListNamedShadowsForThing`

Représente l'autorisation de répertorier les objets nommés Shadows. L'autorisation `iot:ListNamedShadowsForThing` est vérifiée chaque fois qu'une demande est faite pour répertorier un objet nommé Shadows.

`iot:UpdateThingShadow`

Représente l'autorisation de mettre à jour un shadow d'appareil. L'autorisation `iot:UpdateThingShadow` est vérifiée chaque fois qu'une demande est faite pour mettre à jour le contenu Device Shadow d'un objet.

> **Note**
>
> Les actions de stratégie d'exécution de tâche s'appliquent uniquement pour le point de terminaison HTTP TLS. Si vous utilisez le point de terminaison MQTT, vous devez utiliser les actions de stratégie MQTT définies dans cette rubrique.
> Pour un exemple de stratégie d'exécution de tâche qui illustre cela, consultez the section called “Exemple de stratégie d'emploi de base” qui fonctionne avec le protocole MQTT.

Actions AWS IoT Core politiques relatives à l'exécution des tâches

`iotjobsdata:DescribeJobExecution`

Représente l'autorisation de récupérer une exécution de tâche pour un objet donné. L'autorisation `iotjobsdata:DescribeJobExecution` est vérifiée chaque fois qu'une demande est faite d'obtenir une exécution de tâche.

`iotjobsdata:GetPendingJobExecutions`

Représente l'autorisation de récupérer la liste des tâches qui ne sont pas à un statut terminal pour un objet. L'autorisation `iotjobsdata:GetPendingJobExecutions` est vérifiée chaque fois qu'une demande est faite de récupérer la liste.

`iotjobsdata:UpdateJobExecution`

Représente l'autorisation de mettre à jour une exécution de tâche. L'autorisation `iotjobsdata:UpdateJobExecution` est vérifiée chaque fois qu'une demande est faite de mettre à jour l'état d'une exécution de tâche.

`iotjobsdata:StartNextPendingJobExecution`

Représente l'autorisation d'obtenir et de démarrer l'exécution de tâche en attente suivante pour un objet. (C'est-à-dire de mettre à jour une exécution de tâche en la faisant passer du statut QUEUED au statut IN_PROGRESS.) L'autorisation

`iotjobsdata:StartNextPendingJobExecution` est vérifiée chaque fois qu'une demande est faite de démarrer l'exécution de tâche en attente suivante.

AWS IoT Core Action en matière de politique des fournisseurs d'informations d'identification

`iot:AssumeRoleWithCertificate`

Représente l'autorisation AWS IoT Core d'appeler le fournisseur d'informations d'identification pour assumer un rôle IAM avec une authentification basée sur des certificats. L'`iot:AssumeRoleWithCertificate`autorisation est vérifiée chaque fois qu'une demande est faite au fournisseur d' AWS IoT Core informations d'identification pour qu'il assume un rôle.

AWS IoT Core ressources d'action

Pour spécifier une ressource pour une action AWS IoT Core de politique, utilisez l'Amazon Resource Name (ARN) de la ressource. Toutes les ressources ARNs suivent le format suivant :

```
arn:partition:iot:region:AWS-account-ID:Resource-type/Resource-name
```

Le tableau suivant indique la ressource à spécifier pour chaque type d'action. Les exemples d'ARN concernent l'ID du compte`123456789012`, dans la partition`aws`, et sont spécifiques à la région`us-east-1`. Pour plus d'informations sur les formats pour ARNs, consultez Amazon Resource Names (ARNs) dans le guide de AWS Identity and Access Management l'utilisateur.

Action	Type de ressour	Nom de la ressource	Exemples d’ARN
`iot:Connect`	`client`	L'identifiant client du client	`arn:aws:iot:us-east-1:123456789012:client/myClientId`
`iot:DeleteThingShadow`	`thing`	Le nom de l’objet et le nom de l'ombre, le cas échéant	`arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne`

Action	Type de ressour	Nom de la ressource	Exemples d'ARN
iotjobsdata:DescribeJobExecution	thing	Le nom de l'objet	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iotjobsdata:GetPendingJobExecutions	thing	Le nom de l'objet	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetRetainedMessage	topic	Une rubrique de message conservé	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:GetThingShadow	thing	Le nom de l'objet et le nom de l'ombre, le cas échéant	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:ListNamedShadowsForThing	Tous	Tous	*
iot:ListRetainedMessages	Tous	Tous	*
iot:Publish	topic	Une chaîne de rubrique	arn:aws:iot:us-east-1:123456789012:topic/myTopicName

Action	Type de ressour	Nom de la ressource	Exemples d'ARN
iot:Receive	topic	Une chaîne de rubrique	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:RetainPublish	topic	Une rubrique à publier avec l'indicateur CONSERVER défini	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iotjobsdata:StartNextPendingJobExecution	thing	Le nom de l'objet	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:Subscribe	topicf er	Une chaîne de filtre d'objet	arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter
iotjobsdata:UpdateJobExecution	thing	Le nom de l'objet	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:UpdateThingShadow	thing	Le nom de l'objet et le nom de l'ombre, le cas échéant	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:AssumeRoleWithCertificate	rolea]	Un alias de rôle qui pointe vers un rôle ARN	arn:aws:iot:us-east-1:123456789012:rolealias/CredentialProviderRole_alias

AWS IoT Core variables de politique

AWS IoT Core définit les variables de stratégie qui peuvent être utilisées dans AWS IoT Core les politiques du `Condition` bloc `Resource` or. Lorsqu'une stratégie est évaluée, ses variables sont remplacées par des valeurs réelles. Par exemple, si un appareil est connecté au courtier de AWS IoT Core messages avec un ID client 100-234-3456, la variable de politique est remplacée dans le document de `iot:ClientId` politique par 100-234-3456.

AWS IoT Core les politiques peuvent utiliser des caractères génériques et suivre une convention similaire à celle des politiques IAM. L'insertion d'un * (astérisque) dans la chaîne peut être traitée comme un caractère générique, correspondant à n'importe quel caractère. Par exemple, vous pouvez utiliser * pour décrire plusieurs noms de rubriques MQTT dans l'attribut d'une stratégie `Resource`. Les caractères + et # sont traités comme des chaînes littérales dans une stratégie. Pour obtenir un exemple de stratégie montrant comment utiliser les caractères génériques, consultez Utilisation de caractères génériques dans MQTT et les politiques AWS IoT Core.

Vous pouvez également utiliser des variables de stratégie prédéfinies avec des valeurs fixes pour représenter des caractères qui autrement auraient une signification particulière. Ces caractères spéciaux incluent `${*}`, `${?}`, et `${$}`. Pour plus d'informations sur les variables de stratégie et les caractères spéciaux, consultez Éléments de politique IAM : variables et balises et Création d'une condition avec plusieurs clés ou valeurs.

Rubriques

- Variables de AWS IoT Core politique de base
- Variables de stratégie d'objet
- Variables de AWS IoT Core politique de certificat X.509

Variables de AWS IoT Core politique de base

AWS IoT Core définit les variables de politique de base suivantes :

- `aws:SourceIp`: adresse IP du client connecté au courtier de AWS IoT Core messages.
- `iot:ClientId` : ID client utilisé pour se connecter à l'agent de messages AWS IoT Core .
- `iot:DomainName`: nom de domaine du client connecté AWS IoT Core.

Exemples

- Exemples ClientId et variables SourceIp de politique
- Exemples de variable iot:DomainName de politique

Exemples **`ClientId`** et variables **`SourceIp`** de politique

La AWS IoT Core stratégie suivante montre une stratégie qui utilise des variables de stratégie. `aws:SourceIp`peut être utilisé dans l'élément Condition de votre politique pour permettre aux principaux de faire des demandes d'API uniquement dans une plage d'adresses spécifique. Pour obtenir des exemples, consultez Autoriser les utilisateurs et les services cloud à utiliser les tâches AWS IoT.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/clientid1"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
   ],
   "Condition": {
    "IpAddress": {
     "aws:SourceIp": "123.45.167.89"
    }
   }
  }
 ]
```

```
}
```

Dans ces exemples, `${iot:ClientId}` il est remplacé par l'ID du client connecté au courtier de AWS IoT Core messages lorsque la politique est évaluée. Lorsque vous utilisez des variables de stratégie telles que `${iot:ClientId}`, vous pouvez ouvrir par inadvertance l'accès à des rubriques imprévues. Par exemple, si vous utilisez un stratégie qui utilise `${iot:ClientId}` pour spécifier un filtre de rubrique :

```
{
 "Effect": "Allow",
 "Action": [
  "iot:Subscribe"
 ],
 "Resource": [
  "arn:aws:iot:us-east-1:123456789012:topicfilter/my/${iot:ClientId}/topic"
 ]
}
```

Un client peut se connecter en utilisant + comme ID de client. Cela permettrait à l'utilisateur de s'abonner à n'importe quelle rubrique correspondant au filtre de rubrique `my/+/topic`. Pour vous protéger contre de telles failles de sécurité, utilisez l'action de `iot:Connect` stratégie pour contrôler quel client IDs peut se connecter. Par exemple, cette stratégie autorise uniquement les clients dont l'ID client est `clientid1` à se connecter :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/clientid"
   ]
  }
 ]
```

```
}
```

> **Note**
>
> L'utilisation de la variable de la stratégie `${iot:ClientId}` avec `Connect` n'est pas recommandée. Il n'y a pas de contrôle sur la valeur de `ClientId`, donc un attacheur avec un ID client différent peut valider mais provoquer une déconnexion. Étant donné que tout `ClientId` est autorisé, la définition d'un ID client aléatoire peut contourner les politiques de groupe d'objets.

Exemples de variable **`iot:DomainName`** de politique

Vous pouvez ajouter la variable de `iot:DomainName` politique pour limiter les domaines autorisés à utiliser. L'ajout de la variable `iot:DomainName` de politique permet aux appareils de se connecter uniquement à des points de terminaison configurés spécifiques.

La politique suivante permet aux appareils de se connecter au domaine spécifié.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": {
  "Sid": "AllowConnectionsToSpecifiedDomain",
  "Effect": "Allow",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
  "Condition": {
   "StringEquals": {
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
 }
}
```

La politique suivante interdit aux appareils de se connecter au domaine spécifié.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": {
  "Sid": "DenyConnectionsToSpecifiedDomain",
  "Effect": "Deny",
  "Action": [
   "iot:Connect"
  ],
  "Resource": "arn:aws:iot:us-east-1:123456789012:client/clientid",
  "Condition": {
   "StringEquals": {
    "iot:DomainName": "d1234567890abcdefghij-ats.iot.us-east-1.amazonaws.com"
   }
  }
 }
}
```

Pour plus d'informations sur l'opérateur conditionnel de politique, voir Éléments de stratégie IAM JSON : opérateurs de condition. Pour plus d'informations sur les configurations de domaine, voir Qu'est-ce qu'une configuration de domaine ? .

Variables de stratégie d'objet

Les variables de politique d'objet vous permettent d'écrire des AWS IoT Core politiques qui accordent ou refusent des autorisations en fonction des propriétés des objets, telles que les noms des objets, les types d'objets et les valeurs des attributs des objets. Vous pouvez utiliser les variables de politique des objets pour appliquer la même politique afin de contrôler de nombreux AWS IoT Core appareils. Pour plus d'informations sur la mise en service des appareils, veuillez consulter Mise en service des appareils.

Si vous utilisez une association d'objets non exclusive, le même certificat peut être associé à plusieurs objets. Pour maintenir une association claire et éviter les conflits potentiels, vous devez associer votre identifiant client au nom de l'objet. Dans ce cas, vous obtenez le nom de l'objet à partir de l'ID du client dans le `Connect` message MQTT envoyé lorsqu'un objet se connecte à AWS IoT Core.

Gardez ce qui suit à l'esprit lorsque vous utilisez des variables de stratégie d'objet dans les stratégies AWS IoT Core .

- Utilisez l'AttachThingPrincipalAPI pour associer des certificats ou des principaux (identités Amazon Cognito authentifiées) à un objet.
- Si une association d'objets non exclusive est en place, lorsque vous remplacez des noms d'objets par des variables de politique d'objets, la valeur de `clientId` dans le message de connexion MQTT ou dans la connexion TLS doit correspondre exactement au nom de l'objet.

Les variables de stratégie d'objet suivantes sont disponibles :

- `iot:Connection.Thing.ThingName`

  Cela correspond au nom de l'objet dans le AWS IoT Core registre pour lequel la politique est évaluée. AWS IoT Core utilise le certificat présenté par l'appareil lorsqu'il s'authentifie pour déterminer quel élément utiliser pour vérifier la connexion. Cette variable de politique n'est disponible que lorsqu'un appareil se connecte via MQTT ou MQTT via le WebSocket protocole.
- `iot:Connection.Thing.ThingTypeName`

  Cette variable est résolue en type d'objet associé à l'objet pour lequel la stratégie est évaluée. L'ID client de la WebSocket connexion MQTT/ doit être identique au nom de l'objet. Cette variable de politique n'est disponible que lors de la connexion via MQTT ou MQTT via le WebSocket protocole.
- `iot:Connection.Thing.Attributes[`*`attributeName`*`]`

  Cette variable est résolue en valeur d'attribut spécifié associé à l'objet pour lequel la stratégie est évaluée. Un objet peut posséder jusqu'à 50 attributs. Chaque attribut est disponible en tant que variable de politique : `iot:Connection.Thing.Attributes[`*`attributeName`*`]` où *`attributeName`* est le nom de l'attribut ? L'ID client de la MQTT/WebSocket connexion doit être identique au nom de l'objet. Cette variable de politique n'est disponible que lors de la connexion via MQTT ou MQTT via le WebSocket protocole.
- `iot:Connection.Thing.IsAttached`

  `iot:Connection.Thing.IsAttached: ["true"]`impose que seuls les appareils enregistrés AWS IoT et attachés au principal puissent accéder aux autorisations définies dans la politique. Vous pouvez utiliser cette variable pour empêcher un appareil de se connecter AWS IoT Core s'il présente un certificat qui n'est pas associé à un objet IoT dans le AWS IoT Core registre. Cette variable contient des valeurs `true` ou `false` indique que l'objet de connexion est attaché au

certificat ou à l'identité Amazon Cognito dans le registre à l'aide de l'API. AttachThingPrincipal Le nom de l'objet est pris comme identifiant client.

Si votre ID client correspond au nom de votre objet, ou si vous attachez votre certificat à un objet exclusivement, l'utilisation de variables de stratégie dans la définition de la stratégie peut simplifier la gestion des politiques. Au lieu de créer des politiques individuelles pour chaque objet IoT, vous pouvez définir une politique unique à l'aide des variables de politique de l'objet. Cette politique peut être appliquée de manière dynamique à tous les appareils. Voici un exemple de politique illustrant son fonctionnement. Pour de plus amples informations, veuillez consulter ???.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Condition": {
    "StringLike": {
     "iot:ClientId": "*${iot:Connection.Thing.Attributes[envType]}"
    }
   },
   "Effect": "Allow",
   "Action": "iot:Connect",
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/*"
  }
 ]
}
```

Cet exemple de politique permet aux objets de se connecter AWS IoT Core si leur ID client se termine par la valeur de leur envType attribut. Seuls les objets présentant un modèle d'ID client correspondant seront autorisés à se connecter.

Variables de AWS IoT Core politique de certificat X.509

Les variables de politique de certificat X.509 facilitent l'écriture des AWS IoT Core politiques. Ces politiques accordent des autorisations en fonction des attributs du certificat X.509. Les sections suivantes décrivent comment utiliser ces variables de politique de certificat.

> **Important**
>
> Si votre certificat X.509 n'inclut aucun attribut de certificat particulier mais que la variable de politique de certificat correspondante est utilisée dans votre document de stratégie, l'évaluation de la politique peut entraîner un comportement inattendu.

CertificateId

Dans l'RegisterCertificateAPI, le `certificateId` apparaît dans le corps de la réponse. Pour obtenir des informations sur votre certificat, utilisez le `certificateId` in DescribeCertificate.

Attributs de l'émetteur

Les variables AWS IoT Core de politique suivantes permettent d'autoriser ou de refuser des autorisations, en fonction des attributs de certificat définis par l'émetteur du certificat.

- `iot:Certificate.Issuer.DistinguishedNameQualifier`
- `iot:Certificate.Issuer.Country`
- `iot:Certificate.Issuer.Organization`
- `iot:Certificate.Issuer.OrganizationalUnit`
- `iot:Certificate.Issuer.State`
- `iot:Certificate.Issuer.CommonName`
- `iot:Certificate.Issuer.SerialNumber`
- `iot:Certificate.Issuer.Title`
- `iot:Certificate.Issuer.Surname`
- `iot:Certificate.Issuer.GivenName`
- `iot:Certificate.Issuer.Initials`
- `iot:Certificate.Issuer.Pseudonym`
- `iot:Certificate.Issuer.GenerationQualifier`

Attributs de l'objet

Les variables AWS IoT Core de politique suivantes prennent en charge l'octroi ou le refus d'autorisations, en fonction des attributs du sujet du certificat définis par l'émetteur du certificat.

- `iot:Certificate.Subject.DistinguishedNameQualifier`
- `iot:Certificate.Subject.Country`
- `iot:Certificate.Subject.Organization`
- `iot:Certificate.Subject.OrganizationalUnit`
- `iot:Certificate.Subject.State`
- `iot:Certificate.Subject.CommonName`
- `iot:Certificate.Subject.SerialNumber`
- `iot:Certificate.Subject.Title`
- `iot:Certificate.Subject.Surname`
- `iot:Certificate.Subject.GivenName`
- `iot:Certificate.Subject.Initials`
- `iot:Certificate.Subject.Pseudonym`
- `iot:Certificate.Subject.GenerationQualifier`

Les certificats X.509 fournissent à ces attributs la possibilité de contenir une ou plusieurs valeurs. Par défaut, les variables de stratégie de chaque attribut à valeurs multiples renvoient la première valeur. Par exemple, l'attribut `Certificate.Subject.Country` peut contenir une liste de noms de pays, mais `iot:Certificate.Subject.Country` est remplacé par le nom du premier pays lorsqu'il est évalué dans une stratégie.

Vous pouvez demander une valeur d'attribut spécifique autre que la première valeur en utilisant un index de base un. Par exemple, `iot:Certificate.Subject.Country.1` est remplacé par le deuxième nom de pays dans l'attribut `Certificate.Subject.Country`. Si vous spécifiez une valeur d'index qui n'existe pas (par exemple, si vous demandez une troisième valeur alors qu'il n'y a que deux valeurs affectées à l'attribut), aucune substitution n'est effectuée et l'autorisation échoue. Vous pouvez utiliser le suffixe `.List` dans le nom de la variable de stratégie pour spécifier l'ensemble des valeurs de l'attribut.

Attributs de nom alternatif d'émetteur

Les variables de AWS IoT Core politique suivantes prennent en charge l'octroi ou le refus d'autorisations, en fonction des attributs de nom alternatif de l'émetteur définis par l'émetteur du certificat.

- `iot:Certificate.Issuer.AlternativeName.RFC822Name`

- `iot:Certificate.Issuer.AlternativeName.DNSName`
- `iot:Certificate.Issuer.AlternativeName.DirectoryName`
- `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Issuer.AlternativeName.IPAddress`

Attributs de nom alternatif d'objet

Les variables AWS IoT Core de politique suivantes prennent en charge l'octroi ou le refus d'autorisations, en fonction des attributs de nom alternatif du sujet définis par l'émetteur du certificat.

- `iot:Certificate.Subject.AlternativeName.RFC822Name`
- `iot:Certificate.Subject.AlternativeName.DNSName`
- `iot:Certificate.Subject.AlternativeName.DirectoryName`
- `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Subject.AlternativeName.IPAddress`

Autres attributs

Vous pouvez l'utiliser `iot:Certificate.SerialNumber` pour autoriser ou refuser l'accès aux AWS IoT Core ressources, en fonction du numéro de série d'un certificat. La variable de stratégie `iot:Certificate.AvailableKeys` contient le nom de toutes les variables de stratégie de certificat contenant des valeurs.

Utilisation de variables de politique de certificat X.509

Cette rubrique explique en détail comment utiliser les variables de politique de certificat. Les variables de politique de certificat X.509 sont essentielles lorsque vous créez des AWS IoT Core politiques qui accordent des autorisations basées sur les attributs du certificat X.509. Si votre certificat X.509 n'inclut aucun attribut de certificat particulier mais que la variable de politique de certificat correspondante est utilisée dans votre document de stratégie, l'évaluation de la politique peut entraîner un comportement inattendu. Cela est dû au fait que la variable de stratégie manquante n'est pas évaluée dans la déclaration de stratégie.

Dans cette rubrique :

- Exemple de certificat X.509
- Utilisation des attributs de l'émetteur de certificats comme variables de politique de certificat

- Utilisation des attributs du sujet du certificat comme variables de politique de certificat
- Utilisation des attributs de nom alternatif de l'émetteur du certificat comme variables de politique de certificat
- Utilisation des attributs de nom alternatif du sujet du certificat comme variables de politique de certificat
- Utilisation d'un autre attribut de certificat comme variable de politique de certificat
- Limitations applicables aux variables de stratégie de certificat X.509
- Exemples de politiques utilisant des variables de politique de certificat

Exemple de certificat X.509

Un certificat X.509 typique peut apparaître comme suit. Cet exemple de certificat inclut des attributs de certificat. Lors de l'évaluation des AWS IoT Core politiques, les attributs de certificat suivants seront renseignés sous forme de variables de politique de certificat : `Serial Number` `IssuerSubject`,`X509v3 Issuer Alternative Name`,, et`X509v3 Subject Alternative Name`.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            92:12:85:cb:b7:a5:e0:86
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=IoT Devices, OU=SmartHome, ST=WA, CN=IoT Devices Primary CA,
    GN=Primary CA1/initials=XY/dnQualifier=Example corp,
    SN=SmartHome/ title=CA1/pseudonym=Primary_CA/generationQualifier=2/serialNumber=987

        Validity
            Not Before: Mar 26 03:25:40 2024 GMT
            Not After : Apr 28 03:25:40 2025 GMT
        Subject: C=US, O=IoT Devices, OU=LightBulb, ST=NY, CN=LightBulb Device Cert,
    GN=Bulb/initials=ZZ/dnQualifier=Bulb001,
    SN=Multi Color/title=RGB/pseudonym=RGB Device/generationQualifier=4/
serialNumber=123
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    << REDACTED >>
```

```
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:example.com, IP Address:1.2.3.4, URI:ResourceIdentifier001,
email:device1@example.com, DirName:/C=US/O=IoT/OU=SmartHome/CN=LightBulbCert
            X509v3 Issuer Alternative Name:
                DNS:issuer.com, IP Address:5.6.7.8, URI:PrimarySignerCA,
email:primary@issuer.com, DirName:/C=US/O=Issuer/OU=IoT Devices/CN=Primary Issuer CA
    Signature Algorithm: sha256WithRSAEncryption
         << REDACTED >>
```

Utilisation des attributs de l'émetteur de certificats comme variables de politique de certificat

Le tableau suivant fournit des informations détaillées sur la manière dont les attributs de l'émetteur du certificat seront renseignés dans une AWS IoT Core politique.

Attributs de l'émetteur à renseigner dans une politique

Attributs de l'émetteur du certificat	Variables de politique de certificat
• C = NOUS • O = Appareils IoT • UO = SmartHome • ST = WA • CN=IoT Devices (CA principale) • GN = primaire CA1 • Initiales = XY • DNQualifier=Exemple de groupe • SN = SmartHome • titre= CA1 • Pseudonyme = Primary_CA	• `iot:Certificate.Issuer.Country = US` • `iot:Certificate.Issuer.Organization = IoT Devices` • `iot:Certificate.Issuer.OrganizationalUnit = SmartHome` • `iot:Certificate.Issuer.State = WA` • `iot:Certificate.Issuer.CommonName = IoT Devices Primary CA` • `iot:Certificate.Issuer.GivenName = Primary CA1` • `iot:Certificate.Issuer.initials = XY` • `iot:Certificate.Issuer.Distinguished NameQualifier = Example corp`

Attributs de l'émetteur du certificat	Variables de politique de certificat
• Qualificateur de génération = 2 • Numéro de série = 987	• `iot:Certificate.Issuer.Surname = SmartHome` • `iot:Certificate.Issuer.Title = CA1` • `iot:Certificate.Issuer.Pseudonym = Primary_CA` • `iot:Certificate.Issuer.GenerationQualifier = 2` • `iot:Certificate.Issuer.SerialNumber = 987`

Utilisation des attributs du sujet du certificat comme variables de politique de certificat

Le tableau suivant fournit des informations détaillées sur la manière dont les attributs du sujet du certificat seront renseignés dans une AWS IoT Core politique.

Attributs du sujet à renseigner dans une politique

Attributs du sujet du certificat	Variables de politique de certificat
• C = NOUS • O = Appareils IoT • ST = NEW YORK • CN= Certificat de LightBulb périphérique • GN = ampoule • Initiales = ZZ • DNQualifier=BULB001 • SN=Multi Color • Titre=RGB • Pseudonyme = appareil RGB • Qualificateur de génération = 4	• `iot:Certificate.Subject.Country = US` • `iot:Certificate.Subject.Organization = IoT Devices` • `iot:Certificate.Subject.State = NY` • `iot:Certificate.Subject.CommonName = LightBulb Device Cert` • `iot:Certificate.Subject.GivenName = Bulb` • `iot:Certificate.Subject.initials = ZZ` • `iot:Certificate.Subject.DistinguishedNameQualifier = Bulb001` • `iot:Certificate.Subject.Surname = Multi Color` • `iot:Certificate.Subject.Title = RGB` • `iot:Certificate.Subject.Pseudonym = RGB Device`

Attributs du sujet du certificat	Variables de politique de certificat
• Numéro de série = 123	• `iot:Certificate.Subject.GenerationQualifier = 4` • `iot:Certificate.Subject.SerialNumber = 123`

Utilisation des attributs de nom alternatif de l'émetteur du certificat comme variables de politique de certificat

Le tableau suivant fournit des informations détaillées sur la manière dont les attributs de nom alternatif de l'émetteur du certificat seront renseignés dans une AWS IoT Core politique.

Attributs du nom alternatif de l'émetteur à renseigner dans une politique

Nom alternatif de l'émetteur X509v3	Attribut dans une politique
• DNS : issuer.com • Adresse IP : 5.6.7.8 • TYPE : PrimarySigner CA • courriel : primary@issuer.com • DirName: /C= US/O=Issuer/OU=IoT Devices/CN=Émetteur principal CA	• `iot:Certificate.Issuer.AlternativeName.DNSName = issuer.com` • `iot:Certificate.Issuer.AlternativeName.IPAddress = 5.6.7.8` • `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier = PrimarySignerCA` • `iot:Certificate.Issuer.AlternativeName.RFC822Name = primary@issuer.com` • `iot:Certificate.Issuer.AlternativeName.DirectoryName = cn=Primary Issuer CA,ou=IoT Devices,o=Issuer,c=US`

Utilisation des attributs de nom alternatif du sujet du certificat comme variables de politique de certificat

Le tableau suivant fournit des informations détaillées sur la manière dont les attributs du nom alternatif du sujet du certificat seront renseignés dans une AWS IoT Core politique.

Attributs du nom alternatif du sujet à renseigner dans une politique

Nom alternatif du sujet X509v3	Attribut dans une politique
• DNS : Example.com • Adresse IP : 1.2.3.4 • TYPE : ResourceIdentifier 001 • courriel : device1@example.com • DirName:/C= = US/O=IoT/OU=SmartHome/CN LightBulbCert	• `iot:Certificate.Subject.AlternativeName.DNSName = example.com` • `iot:Certificate.Subject.AlternativeName.IPAddress = 1.2.3.4` • `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier = ResourceIdentifier001` • `iot:Certificate.Subject.AlternativeName.RFC822Name = device1@example.com` • `iot:Certificate.Subject.AlternativeName.DirectoryName = cn=LightBulbCert,ou=SmartHome,o=IoT,c=US`

Utilisation d'un autre attribut de certificat comme variable de politique de certificat

Le tableau suivant fournit des informations détaillées sur la manière dont les autres attributs de certificat seront renseignés dans une AWS IoT Core politique.

Autres attributs à renseigner dans une politique

Autre attribut de certificat	Variable de politique de certificat
`Serial Number: 92:12:85:cb:b7:a5:e0:86`	`iot:Certificate.SerialNumber = 105256223 89124227206`

Limitations applicables aux variables de stratégie de certificat X.509

Les limitations suivantes s'appliquent aux variables de stratégie de certificat X.509 :

Variables de stratégie manquantes

Si votre certificat X.509 n'inclut aucun attribut de certificat particulier mais que la variable de politique de certificat correspondante est utilisée dans votre document de stratégie, l'évaluation

de la politique peut entraîner un comportement inattendu. Cela est dû au fait que la variable de stratégie manquante n'est pas évaluée dans la déclaration de stratégie.

SerialNumber Format du certificat

AWS IoT Core traite le numéro de série du certificat comme la représentation sous forme de chaîne d'un entier décimal. Par exemple, si une politique autorise uniquement les connexions dont l'ID client correspond au numéro de série du certificat, l'ID client doit être le numéro de série au format décimal.

Caractères génériques

Si des caractères génériques sont présents dans les attributs du certificat, la variable de politique n'est pas remplacée par la valeur de l'attribut du certificat. Cela laissera le `${policy-variable}` texte dans le document de politique. Cela risque de provoquer un échec d'autorisation. Les caractères génériques suivants peuvent être utilisés : `*`, `$`, `+`, `?` et `#`.

Champs de tableau

Les attributs de certificats qui contiennent des tableaux sont limités à cinq éléments. Les autres éléments sont ignorés.

String length

Toutes les valeurs de chaîne sont limitées à 1 024 caractères. Si un attribut de certificat contient une chaîne de plus de 1024 caractères, la variable de politique n'est pas remplacée par la valeur de l'attribut de certificat. Cela les conservera `${policy-variable}` dans le document de politique. Cela risque de provoquer un échec d'autorisation.

Caractères spéciaux

Tout caractère spécial, tel que `,`, `'`, `\`, `+`, `=`, `<`, `>` et `;` doit être préfixé par une barre oblique inverse (`\`) lorsqu'il est utilisé dans une variable de stratégie. Par exemple, `Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US` devient `Amazon Web Service O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US`.

Exemples de politiques utilisant des variables de politique de certificat

Le document de politique suivant autorise les connexions avec un ID client correspondant au numéro de série du certificat et la publication sur le sujet correspondant au modèle :`${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*`.

> **Important**
>
> Si votre certificat X.509 n'inclut aucun attribut de certificat particulier mais que la variable de politique de certificat correspondante est utilisée dans votre document de stratégie, l'évaluation de la politique peut entraîner un comportement inattendu. Cela est dû au fait que la variable de stratégie manquante n'est pas évaluée dans la déclaration de stratégie. Par exemple, si vous joignez le document de politique suivant à un certificat qui ne contient pas l'`iot:Certificate.Subject.Organization`attribut, les variables de politique de `iot:Certificate.Subject.Organization` certificat ne seront pas renseignées lors de l'évaluation de la politique.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Certificate.SerialNumber}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.Organization}/device-stats/${iot:ClientId}/*"
   ]
  }
 ]
}
```

Vous pouvez également utiliser l'opérateur de condition nulle pour vous assurer que les variables de politique de certificat utilisées dans une politique sont renseignées lors de l'évaluation de la politique. Le document de politique suivant autorise `iot:Connect` les certificats uniquement lorsque les attributs du numéro de série du certificat et du nom commun de l'objet du certificat sont présents.

Toutes les variables de politique de certificat ont des valeurs de chaîne, de sorte que tous les opérateurs de condition de chaîne sont pris en charge.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/*"
   ],
   "Condition": {
    "Null": {
     "iot:Certificate.SerialNumber": "false",
     "iot:Certificate.Subject.CommonName": "false"
    }
   }
  }
 ]
}
```

Prévention du cas de figure de l’adjoint désorienté entre services

Le problème de l’adjoint confus est un problème de sécurité dans lequel une entité qui n’a pas l’autorisation d’effectuer une action peut contraindre une entité plus privilégiée à effectuer cette action. En AWS, l'usurpation d'identité interservices peut entraîner la confusion des adjoints. L’usurpation d’identité entre services peut se produire lorsqu’un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé pour utiliser ses autorisations afin d'agir sur les ressources d'un autre client de sorte qu'il n'y aurait pas accès

autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services avec des principaux de service qui ont eu accès aux ressources de votre compte.

Pour limiter les autorisations qui AWS IoT confèrent un autre service à la ressource, nous vous recommandons d'utiliser les clés de contexte de condition `aws:SourceAccount`globale `aws:SourceArn`et les clés contextuelles dans les politiques de ressources. Si vous utilisez les deux clés de contexte de condition globale, la valeur `aws:SourceAccount` et le compte de la valeur `aws:SourceArn` doit utiliser le même ID de compte lorsqu'il est utilisé dans la même déclaration de stratégie.

Le moyen le plus efficace de se protéger contre le problème de l'adjoint confus est d'utiliser la clé de `aws:SourceArn` contexte de condition globale avec le nom de ressource Amazon (ARN) complet de la ressource. En AWS IoT effet, vous `aws:SourceArn` devez respecter le format : `arn:`*`aws`*`:iot:`*`region`*`:`*`account-id`*`:`*`resource-type/resource-id`* pour les autorisations spécifiques aux ressources ou`arn:`*`aws`*`:iot:`*`region`*`:`*`account-id`*`:`*`*. L'identifiant de ressource peut être le nom ou l'ID de la ressource autorisée, ou une déclaration générique de la ressource autorisée. IDs Assurez-vous que cela *`region`* correspond à votre AWS IoT région et à *`account-id`* votre numéro de compte client.

L'exemple suivant montre comment éviter le problème de confusion des adjoints en utilisant les clés de contexte `aws:SourceArn` et de condition `aws:SourceAccount` globale dans la politique de confiance dans les AWS IoT rôles. Pour obtenir plus d'exemples, consultez Exemples détaillés de prévention de la confusion chez les adjoints.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            },
            "ArnLike":{
```

```
                "aws:SourceArn":"arn:aws:iot:us-east-1:123456789012:*"
            }
        }
    }
  ]
}
```

> **Note**
>
> Si vous recevez des erreurs de refus d'accès, cela peut être dû au fait que l'intégration du service au AWS Security Token Service (STS) ne prend pas en charge les clés de `aws:SourceAccount` contexte `aws:SourceArn` et.

Exemples détaillés de prévention de la confusion chez les adjoints

Cette section fournit des exemples détaillés de la manière de prévenir le problème de confusion des adjoints en utilisant les clés de contexte aws:SourceArn et les clés de contexte de condition aws:SourceAccount globale de la politique de confiance dans les AWS IoT rôles.

- Mise en service de flotte
- JITP
- Fournisseur d'informations d'identification

Mise en service de flotte

Vous pouvez configurer le provisionnement du parc à l'aide d'une ressource de modèle de provisionnement. Lorsqu'un modèle de provisionnement fait référence à un rôle de provisionnement, la politique de confiance de ce rôle peut inclure les clés de `aws:SourceAccount` condition `aws:SourceArn` et. Ces clés limitent les ressources pour lesquelles la configuration peut invoquer la `sts:AssumeRole` demande.

Le rôle associé à la politique de confiance suivante ne peut être assumé que par le principal IoT (`iot.amazonaws.com`) pour le modèle de provisionnement spécifié dans le`SourceArn`.

JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:provisioningtemplate/example_template"
            }
         }
      }
   ]
}
```

JITP

Dans le cadre du just-in-time provisionnement (JITP), vous pouvez soit utiliser le modèle de provisionnement en tant que ressource distincte de l'autorité de certification, soit définir le corps du modèle et le rôle dans le cadre de la configuration du certificat de l'autorité de certification. La valeur de la politique de confiance `aws:SourceArn` in the AWS IoT role dépend de la manière dont vous définissez le modèle de provisionnement.

Définition d'un modèle de provisionnement en tant que ressource distincte

Si vous définissez votre modèle de provisionnement en tant que ressource distincte, la valeur de `aws:SourceArn` peut être`"arn:aws:iot:region:account-id:provisioningtemplate/example_template"`. Vous pouvez utiliser le même exemple de politique dansMise en service de flotte.

Définition d'un modèle de provisionnement dans un certificat CA

Si vous définissez votre modèle de provisionnement dans une ressource de certificat CA, la valeur de `aws:SourceArn` peut être `"arn:aws:iot:region:account-id:cacert/cert_id"` ou`"arn:aws:iot:region:account-id:cacert/*"`. Vous pouvez utiliser un caractère

générique lorsque l'identifiant de la ressource, tel que l'ID d'un certificat CA, est inconnu au moment de la création.

Le rôle associé à la politique de confiance suivante ne peut être assumé que par le principal IoT (`iot.amazonaws.com`) pour le certificat CA spécifié dans le`SourceArn`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"iot.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            },
            "ArnLike":{
               "aws:SourceArn":"arn:aws:iot:us-
east-1:123456789012:cacert/
8ecde6884f3d87b1125ba31ac3fcb13d7016de7f57cc904fe1cb97c6ae98196e"
            }
         }
      }
   ]
}
```

Lorsque vous créez un certificat CA, vous pouvez faire référence à un rôle de provisionnement dans la configuration d'enregistrement. La politique de confiance du rôle de provisionnement peut être utilisée `aws:SourceArn` pour limiter les ressources pour lesquelles le rôle peut être assumé. Toutefois, lors de l'CACertificateappel initial à Register pour enregistrer le certificat CA, vous n'auriez pas l'ARN du certificat CA à spécifier dans la `aws:SourceArn` condition.

Pour contourner ce problème, c'est-à-dire pour spécifier la politique de confiance du rôle d'approvisionnement pour le certificat CA spécifique enregistré auprès de celui-ci AWS IoT Core, vous pouvez effectuer les opérations suivantes :

- Tout d'abord, appelez Register CACertificate sans fournir le `RegistrationConfig` paramètre.
- Une fois le certificat CA enregistré auprès de celui-ci AWS IoT Core, appelez Update CACertificate dessus.

  Dans l'CACertificate appel de mise à jour, fournissez une `RegistrationConfig` politique de confiance incluant le rôle d'approvisionnement, `aws:SourceArn` définie sur l'ARN du certificat CA récemment enregistré.

Fournisseur d'informations d'identification

Pour le fournisseur AWS IoT Core d'informations d'identification, utilisez le même Compte AWS que celui que vous utilisez pour créer l'alias de rôle dans `aws:SourceAccount` et spécifiez une instruction qui correspond à l'ARN de ressource du type de ressource rolealias dans. `aws:SourceArn` Lorsque vous créez un rôle IAM à utiliser avec un fournisseur AWS IoT Core d'informations d'identification, vous devez inclure dans la `aws:SourceArn` condition tous les alias ARNs de rôle susceptibles de devoir assumer le rôle, autorisant ainsi la demande interservices. `sts:AssumeRole`

Le rôle soumis à la politique de confiance suivante ne peut être assumé que par le principal du fournisseur d' AWS IoT Core informations d'identification (`credentials.iot.amazonaws.com`) pour les RoleAlias spécifiés dans le. `SourceArn` Si un principal tente de récupérer les informations d'identification pour un alias de rôle autre que celui spécifié dans la `aws:SourceArn` condition, la demande sera refusée, même si cet autre alias de rôle fait référence au même rôle IAM.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
```

```
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iot:us-
east-1:123456789012:rolealias/example_rolealias"
        }
      }
    }
  ]
}
```

AWS IoT Core exemples de politiques

Les exemples de stratégies de cette section illustrent les documents utilisés pour effectuer les tâches courantes dans AWS IoT Core. Vous pouvez les utiliser comme exemples de départ lors de la création des politiques de vos solutions.

Les exemples de cette section utilisent ces éléments de stratégie :

- the section called “AWS IoT Core actions politiques”
- the section called “AWS IoT Core ressources d'action”
- the section called “Exemples de politiques basées sur l’identité”
- the section called “Variables de AWS IoT Core politique de base”
- the section called “Variables de AWS IoT Core politique de certificat X.509”

Exemples de stratégie dans cette section :

- Exemples de stratégies de connexion
- Exemples de stratégie de publication/abonnement
- Exemples de stratégies de connexion et de publication
- Exemples de stratégies de messages conservés
- Exemples de stratégies de certificat
- Exemples de stratégies d'objet
- Exemple de stratégie d'emploi de base

Exemples de stratégies de connexion

La politique suivante refuse l'autorisation au client IDs `client1` et `client2` à la connexion AWS IoT Core, tout en autorisant les appareils à se connecter à l'aide d'un identifiant client. L'ID client

correspond au nom d'un objet enregistré dans le AWS IoT Core registre et attaché au principal utilisé pour la connexion :

> **Note**
>
> Pour les appareils enregistrés, nous vous recommandons d'utiliser des variables de stratégie d'objet pour les `Connect` actions et d'attacher l'objet au principal utilisé pour la connexion.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1",
    "arn:aws:iot:us-east-1:123456789012:client/client2"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  }
 ]
}
```

La politique suivante accorde l'autorisation de se connecter à l' AWS IoT Core aide de l'ID client`client1`. Cet exemple de stratégie concerne les appareils non enregistrés.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/client1"
   ]
  }
 ]
}
```

Exemples de stratégies de sessions persistantes MQTT

`connectAttributes` vous permettent de spécifier les attributs que vous souhaitez utiliser dans votre message de connexion dans vos politiques IAM telles que `PersistentConnect` et `LastWill`. Pour de plus amples informations, veuillez consulter Utilisation de ConnectAttributes.

La stratégie suivante autorise la connexion avec la fonctionnalité`PersistentConnect` :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
```

```
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
}
```

La stratégie suivante interdit `PersistentConnect`, d'autres fonctionnalités sont autorisées :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringNotEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
}
```

La stratégie ci-dessus peut également être exprimée en utilisant `StringEquals`, toute autre fonctionnalité, y compris une nouvelle fonctionnalité, est autorisée :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "*",
   "Condition": {
    "ForAnyValue:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
    }
   }
  }
 ]
}
```

La stratégie suivante autorise la connexion à la fois par `PersistentConnect` et `LastWill`, toute autre nouvelle fonctionnalité n'est pas autorisée :

JSON

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

```
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
     ]
    }
   }
  }
 ]
}
```

La stratégie suivante autorise une connexion propre par les clients avec ou sans `LastWill`, aucune autre fonctionnalité ne sera autorisée :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
                }
            }
        }
    ]
}
```

La stratégie suivante autorise uniquement la connexion à l'aide des fonctionnalités par défaut :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": []
    }
   }
  }
 ]
}
```

La stratégie suivante autorise la connexion uniquement avec `PersistentConnect`, toute nouvelle fonctionnalité est autorisée tant que la connexion utilise `PersistentConnect`:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
```

```
                    ]
                }
            }
        }
    ]
}
```

La stratégie suivante stipule que la connexion doit avoir à la fois une utilisation `PersistentConnect` et `LastWill`, aucune nouvelle fonctionnalité n'est autorisée :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect",
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "*",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "PersistentConnect"
     ]
```

```
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "*",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Connect"
   ],
   "Resource": "*",
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": []
    }
   }
  }
 ]
}
```

La stratégie suivante ne doit pas avoir `PersistentConnect` mais peut avoir `LastWill`, toute autre nouvelle fonctionnalité n'est pas autorisée :

JSON

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

```
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
                }
            }
        }
    ]
}
```

La stratégie suivante autorise la connexion uniquement aux clients disposant de `LastWill` avec une une rubrique"`my/lastwill/topicName`", toute fonctionnalité est autorisée tant qu'elle utilise la rubrique `LastWill`:

JSON

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

```
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                    "iot:LastWillTopic": "arn:aws:iot:region:account-id:topic/my/
lastwill/topicName"
                }
            }
        }
    ]
}
```

La stratégie suivante autorise uniquement une connexion propre à l'aide d'un fichier `LastWillTopic` spécifique, toute fonctionnalité est autorisée tant qu'elle utilise le `LastWillTopic`:

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
            "Condition": {
                "ArnEquals": {
                    "iot:LastWillTopic": "arn:aws:iot:region:account-id:topic/my/
lastwill/topicName"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        }
    ]
}
```

Exemples de stratégie de publication/abonnement

La politique que vous utilisez dépend de la façon dont vous vous connectez AWS IoT Core. Vous pouvez vous connecter à AWS IoT Core l'aide d'un client MQTT, HTTP ou WebSocket. Lorsque vous vous connectez à un client MQTT, vous vous authentifiez avec un certificat X.509. Lorsque vous vous connectez via HTTP ou le WebSocket protocole, vous vous authentifiez avec Signature Version 4 et Amazon Cognito.

> **Note**
>
> Pour les appareils enregistrés, nous vous recommandons d'utiliser des variables de stratégie d'objet pour les `Connect` actions et d'attacher l'objet au principal utilisé pour la connexion.

Dans cette section :

- Utilisation de caractères génériques dans MQTT et les politiques AWS IoT Core
- Politiques relatives à la publication, à l'abonnement et à la réception de messages sur to/from des sujets spécifiques
- Politiques relatives à la publication, à l'abonnement et à la réception de to/from sujets de messages avec un préfixe spécifique
- Politiques relatives à la publication, à l'abonnement et à la réception de messages (to/from sujets spécifiques à chaque appareil)
- Politiques relatives à la publication, à l'abonnement et à la réception de messages (to/from sujets avec un attribut thing dans le nom du sujet)
- Politiques pour refuser la publication de messages dans les sous-thèmes d'un nom de la rubrique
- Politiques pour refuser la réception de messages provenant de sous-rubriques d'un nom d’objet

- Politiques d'abonnement à des rubriques utilisant des caractères génériques MQTT
- Politiques pour HTTP et les WebSocket clients

Utilisation de caractères génériques dans MQTT et les politiques AWS IoT Core

Le MQTT et AWS IoT Core les politiques ont des caractères génériques différents et vous devez les choisir après mûre réflexion. Dans MQTT, les caractères génériques + et C # sont utilisés dans les filtres de sujets MQTT pour s'abonner à plusieurs noms de sujets. AWS IoT Core les politiques utilisent * et ? comme caractères génériques et respectent les conventions des politiques IAM. Dans un document de stratégie, le * représente n'importe quelle combinaison de caractères et un point d'interrogation ? représente n'importe quel caractère unique. Dans les documents de stratégie, les caractères génériques MQTT + et # sont traités comme des caractères sans signification particulière. Pour décrire plusieurs noms de rubrique et filtres de rubrique dans l'attribut d'une stratégie `resource`, utilisez les caractères génériques * et ? à la place des caractères génériques MQTT.

Lorsque vous choisissez les caractères génériques à utiliser dans un document de politique, considérez que le * caractère n'est pas limité à un seul niveau de sujet. Le + personnage est limité à un seul niveau de sujet dans un filtre de sujet MQTT. Pour limiter une spécification générique à un seul niveau de filtre de rubrique MQTT, envisagez d'utiliser plusieurs caractères ?. Pour plus d'informations sur l'utilisation de caractères génériques dans une ressource de politique et d'autres exemples de leurs correspondances, consultez la section Utilisation de caractères génériques dans une ressource. ARNs

Le tableau ci-dessous montre les différents caractères génériques utilisés dans MQTT et les politiques AWS IoT Core pour les clients MQTT.

Caractère générique	Est-ce un caractère générique MQTT	Exemple dans MQTT	Est-ce un caractère générique de la AWS IoT Core politique	Exemple de AWS IoT Core politiques pour les clients MQTT
#	Oui	`some/#`	Non	N/A

Caractère générique	Est-ce un caractère générique MQTT	Exemple dans MQTT	Est-ce un caractère générique de la AWS IoT Core politique	Exemple de AWS IoT Core politiques pour les clients MQTT
+	Oui	`some/+/topic`	Non	N/A

*	Non	N/A	Oui	`topicfilter/some/*/topic`
				
				`topicfilter/some/sensor*/topic`

?	Non	N/A	Oui	`topic/some/?????/topic`
				
				`topicfilter/some/sensor???/topic`

Politiques relatives à la publication, à l'abonnement et à la réception de messages sur to/from des sujets spécifiques

Vous trouverez ci-dessous des exemples d'appareils enregistrés et non enregistrés pour publier, s'abonner et recevoir des messages dont to/from le sujet est intitulé « some_specific_topic ». Les exemples soulignent également cela `Publish` et `Receive` utilisent « topic » comme ressource, ainsi que « topicfilter » comme ressource.

Registered devices

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter avec un ClientiD qui correspond au nom d'un objet dans le registre. Il fournit également des autorisations `Publish`, `Subscribe` et des autorisations `Receive` pour l’objet nommé « some_special_topic ».

JSON

```
{
```

```
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/some_specific_topic"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
   ]
  }
```

```
 ]
}
```

Unregistered devices

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter à l'aide de ClientID1, ClientID2 ou ClientID3. Il fournit également des autorisations `Publish`, `Subscribe` et des autorisations `Receive` pour l'objet nommé « some_special_topic ».

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/
some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        }
    ]
}
```

Politiques relatives à la publication, à l'abonnement et à la réception de to/from sujets de messages avec un préfixe spécifique

Vous trouverez ci-dessous des exemples d'appareils enregistrés et non enregistrés permettant de publier, de s'abonner et de recevoir des messages dont le préfixe to/from est « topic_prefix ».

> **Note**
>
> Notez l'utilisation du caractère générique * dans cet exemple. Bien qu'il * soit utile de fournir des autorisations pour plusieurs noms de sujets dans une seule déclaration, cela peut avoir des conséquences imprévues en octroyant plus de privilèges aux appareils que ce qui est nécessaire. Nous vous recommandons donc de n'utiliser le caractère générique qu'*après mûre réflexion.

Registered devices

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter avec un ClientiD qui correspond au nom d'un objet dans le registre. Il fournit également des autorisations `Publish`, `Subscribe` et des autorisations `Receive` pour les sujets préfixés par « topic_prefix ».

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
   ]
  }
 ]
}
```

Unregistered devices

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter à l'aide de ClientID1, ClientID2 ou ClientID3. Il fournit également des autorisations `Publish`, `Subscribe` et des autorisations `Receive` pour les sujets préfixés par « topic_prefix ».

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
            ]
        }
```

```
    ]
}
```

Politiques relatives à la publication, à l'abonnement et à la réception de messages (to/from sujets spécifiques à chaque appareil)

Vous trouverez ci-dessous des exemples d'appareils enregistrés et non enregistrés permettant de publier, de s'abonner et de recevoir des messages to/from sur des sujets spécifiques à l'appareil en question.

Registered devices

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter avec un ClientiD qui correspond au nom d'un objet dans le registre. Il donne l'autorisation de publier sur de rubrique spécifique à l'objet (`sensor/device/${iot:Connection.Thing.ThingName}`), ainsi que de s'abonner et de recevoir du sujet spécifique à l'objet (`command/device/${iot:Connection.Thing.ThingName}`). Si le nom de l'objet dans le registre est « thing1", l'appareil pourra publier dans le sujet « sensor/device/thing1". The device will also be able to subscribe to and receive from the topic "command/device/thing 1".

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
```

```
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
   ]
  }
 ]
}
```

Unregistered devices

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter à l'aide de ClientID1, ClientID2 ou ClientID3. Il donne l'autorisation de publier sur de rubrique spécifique au client (`sensor/device/${iot:ClientId}`), ainsi que de s'abonner et de recevoir de rubrique spécifique au client (`command/device/${iot:ClientId}`). Si l'appareil se connecte avec ClientID en tant que ClientID1, il pourra

publier sur le sujet « 1". sensor/device/clientId L'appareil pourra également s'abonner au sujet et en recevoir`device/clientId1/command`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/command/
device/${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
```

```
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

Politiques relatives à la publication, à l'abonnement et à la réception de messages (to/from sujets avec un attribut thing dans le nom du sujet)

Vous trouverez ci-dessous un exemple permettant aux appareils enregistrés de publier, de s'abonner et de recevoir des to/from sujets de messages dont le nom inclut des attributs d'objet.

> **Note**
>
> Les attributs d'objet n'existent que pour les appareils enregistrés dans AWS IoT Core le registre. Il n'existe pas d'exemple correspondant pour les appareils non enregistrés.

Registered devices

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter avec un ClientiD qui correspond au nom d'un objet dans le registre. Il donne l'autorisation de publier sur la rubrique (`sensor/${iot:Connection.Thing.Attributes[version]}`), de s'abonner et de recevoir de la rubrique (`command/${iot:Connection.Thing.Attributes[location]}`) où le nom inclut des attributs d'objet. Si le nom de l'objet dans le registre contient un `version=v1` et`location=Seattle`, l'appareil pourra publier dans le sujet « sensor/v1", and subscribe to and receive from the topic "command/Seattle ».

JSON

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

```
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/sensor/
${iot:Connection.Thing.Attributes[version]}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Subscribe"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topicfilter/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Receive"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/command/
${iot:Connection.Thing.Attributes[location]}"
   ]
  }
 ]
```

```
}
```

Unregistered devices

Comme les attributs d'objet n'existent que pour les appareils enregistrés dans AWS IoT Core le registre, il n'existe aucun exemple correspondant pour les objets non enregistrés.

Politiques pour refuser la publication de messages dans les sous-thèmes d'un nom de la rubrique

Ce qui suit montre des exemples d'appareils enregistrés et non enregistrés pour publier des messages sur toutes les rubriques, à l'exception de certaines sous-rubriques.

Registered devices

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter avec un ClientiD qui correspond au nom d'un objet dans le registre. Il donne l'autorisation de publier sur tous les rubriques précédées du préfixe « department/ », mais pas sur la sous-rubrique « department/admins ».

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
```

```
   "Effect": "Allow",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/department/*"
   ]
  },
  {
   "Effect": "Deny",
   "Action": [
    "iot:Publish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
   ]
  }
 ]
}
```

Unregistered devices

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter à l'aide de ClientID1, ClientID2 ou ClientID3. Il donne l'autorisation de publier sur tous les rubriques précédées du préfixe « department/ », mais pas sur la sous-rubrique « department/admins ».

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
            ]
        }
    ]
}
```

Politiques pour refuser la réception de messages provenant de sous-rubriques d'un nom d’objet

Ce qui suit montre des exemples d'appareils enregistrés et non enregistrés pour s'abonner et recevoir des messages de rubriques avec des préfixes spécifiques, à l'exception de certains sous-rubriques.

Registered devices

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter avec un ClientiD qui correspond au nom d'un objet dans le registre. La stratégie permet aux appareils de s'abonner à n'importe quel sujet portant le préfixe « topic_prefix ». `NotResource`En utilisant dans l'instruction for`iot:Receive`, nous autorisons l'appareil à recevoir des messages provenant de tous les sujets auxquels il est abonné, à l'exception des sujets préfixés par « prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted", however, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restricted topic_ ».

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix/*"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Receive",
   "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix/
restricted/*"
  }
 ]
}
```

Unregistered devices

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter à l'aide de ClientID1, ClientID2 ou ClientID3. La stratégie permet aux appareils de s'abonner à n'importe quel sujet portant le préfixe « topic_prefix ». `NotResource`En utilisant dans l'instruction for`iot:Receive`, nous autorisons l'appareil à recevoir des messages relatifs à tous les sujets auxquels il est abonné, à l'exception des

sujets préfixés par « prefix/restricted". For example, with this policy, devices can subscribe to "topic_prefix/topic1" and even "topic_prefix/restricted". However, they will only receive messages from the topic "topic_prefix/topic1" and no messages from the topic "topic_prefix/restricted topic_ ».

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/
topic_prefix/*"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/
topic_prefix/restricted/*"
        }
    ]
}
```

Politiques d'abonnement à des rubriques utilisant des caractères génériques MQTT

Les caractères génériques MQTT + et # sont traités comme des chaînes littérales, mais ils ne le sont pas lorsqu'ils sont utilisés dans des politiques. AWS IoT Core Dans MQTT, + et # sont traités comme des caractères génériques uniquement lors de l'abonnement à un filtre d’objet, mais comme une

chaîne littérale dans tous les autres contextes. Nous vous recommandons de n'utiliser ces caractères génériques MQTT que dans le cadre de AWS IoT Core politiques après mûre réflexion.

Vous trouverez ci-dessous des exemples d'objets enregistrés et non enregistrés utilisant des caractères génériques MQTT dans les politiques. AWS IoT Core Ces caractères génériques sont traités comme des chaînes littérales.

Registered devices

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter avec un ClientiD qui correspond au nom d'un objet dans le registre. La stratégie permet aux appareils de s'abonner aux rubriques « département/+/employés » et « emplacement/# ». Dans la mesure où + et # sont traités comme des chaînes littérales dans AWS IoT Core les politiques, les appareils peuvent s'abonner à la rubrique « département/+/employés », mais pas à la rubrique « ». department/engineering/employees". Similarly, devices can subscribe to the topic "location/#" but not to the topic "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
   ],
   "Condition": {
    "Bool": {
     "iot:Connection.Thing.IsAttached": "true"
    }
   }
  },
  {
```

```
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/department/+/
employees"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Subscribe",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/#"
  },
  {
   "Effect": "Allow",
   "Action": "iot:Receive",
   "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
  }
 ]
}
```

Unregistered devices

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante permet aux appareils de se connecter à l'aide de ClientID1, ClientID2 ou ClientID3. La stratégie permet aux appareils de s'abonner aux rubriques « département/+/employés » et « emplacement/# ». Dans la mesure où + et # sont traités comme des chaînes littérales dans AWS IoT Core les politiques, les appareils peuvent s'abonner à la rubrique « département/+/employés », mais pas à la rubrique « ». department/engineering/employees". Similarly, devices can subscribe to the topic "location/#" but not "location/Seattle". However, once the device subscribes to the topic "department/+/employees", the policy will allow them to receive messages from the topic "department/engineering/employees". Similarly, once the device subscribes to the topic "location/#", they will receive messages from the topic "location/Seattle

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
```

```
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/
department/+/employees"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/
location/#"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
        }
    ]
}
```

Politiques pour HTTP et les WebSocket clients

Lorsque vous vous connectez via HTTP ou le WebSocket protocole, vous vous authentifiez avec Signature Version 4 et Amazon Cognito. Les identités Amazon Cognito peuvent être authentifiées ou non. Les identités authentifiées appartiennent aux utilisateurs authentifiés par tout fournisseur d'identité pris en charge. Les identités non authentifiées appartiennent généralement à des utilisateurs invités qui ne s'authentifient pas auprès d'un fournisseur d'identité. Amazon Cognito fournit un identifiant unique et des informations d' AWS identification pour prendre en charge les identités non authentifiées. Pour de plus amples informations, veuillez consulter the section called “Autorisation avec les identités Amazon Cognito”.

Pour les opérations suivantes, AWS IoT Core utilise des AWS IoT Core politiques associées aux identités Amazon Cognito via l'`AttachPolicy`API. Cela permet de définir les autorisations associées au pool d'identités Amazon Cognito avec des identités authentifiées.

- `iot:Connect`
- `iot:Publish`
- `iot:Subscribe`
- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Cela signifie qu'une identité Amazon Cognito doit être autorisée par la politique de rôle IAM et la politique. AWS IoT Core Vous attachez la politique de rôle IAM au pool et la AWS IoT Core politique à Amazon Cognito Identity via AWS IoT Core `AttachPolicy` l'API.

Les utilisateurs authentifiés et non authentifiés sont des types d'identité différents. Si vous n'associez aucune AWS IoT politique à l'identité Amazon Cognito, un utilisateur authentifié ne parvient pas à s'authentifier AWS IoT et n'a pas accès aux AWS IoT ressources et aux actions.

> **Note**
>
> Pour les autres AWS IoT Core opérations ou pour les identités non authentifiées, AWS IoT Core ne limite pas les autorisations associées au rôle de pool d'identités Amazon Cognito. Pour les identités authentifiées et non authentifiées, c'est la stratégie la plus permissive que nous vous recommandons d'attacher au rôle de réserve d'identités Amazon Cognito.

HTTP

Pour autoriser les identités Amazon Cognito non authentifiées à publier des messages via HTTP sur une rubrique spécifique à l'identité Amazon Cognito, attachez la politique IAM suivante au rôle de réserve d'identités Amazon Cognito :

JSON

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

```
            "Action": [
                "iot:Publish",
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        }
    ]
}
```

Pour autoriser les utilisateurs authentifiés, associez la politique précédente au rôle de pool Amazon Cognito Identity et à Amazon Cognito Identity à l'aide de l'API. AWS IoT Core AttachPolicy

> **Note**
>
> Lorsque vous autorisez les identités Amazon Cognito AWS IoT Core , prenez en compte les deux politiques et accordez le minimum de privilèges spécifié. Une action n'est autorisée que si les deux stratégies autorisent l'action demandée. Si l'une des politiques empêche une action, cette action n'est pas autorisée.

MQTT

Pour autoriser les identités Amazon Cognito non authentifiées à publier des messages MQTT WebSocket sur un sujet spécifique à l'identité Amazon Cognito de votre compte, associez la politique IAM suivante au rôle du pool d'identités Amazon Cognito :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${cognito-
identity.amazonaws.com:sub}"]
        }
    ]
}
```

Pour autoriser les utilisateurs authentifiés, associez la politique précédente au rôle de pool Amazon Cognito Identity et à Amazon Cognito Identity à l'aide de l'API. AWS IoT Core AttachPolicy

> **Note**
>
> Lorsque vous autorisez les identités Amazon Cognito AWS IoT Core , prenez en compte les deux et accordez le minimum de privilèges spécifié. Une action n'est autorisée que si les deux stratégies autorisent l'action demandée. Si l'une des politiques empêche une action, cette action n'est pas autorisée.

Exemples de stratégies de connexion et de publication

Pour les appareils enregistrés en tant qu'objets dans le AWS IoT Core registre, la politique suivante autorise la connexion à l' AWS IoT Core aide d'un ID client correspondant au nom de l'objet et limite le périphérique à la publication sur un sujet MQTT spécifique à un ID client ou à un nom d'objet. Pour qu'une connexion soit réussie, le nom de l'objet doit être enregistré dans le AWS IoT Core registre et authentifié à l'aide d'une identité ou d'un principal attaché à l'objet :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action":["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}"]
      },
```

```
      {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
      }
    ]
}
```

Pour les appareils qui ne sont pas enregistrés en tant qu'objets dans le AWS IoT Core registre, la politique suivante autorise la connexion à AWS IoT Core l'aide de l'ID client `client1` et limite l'appareil à publier sur un sujet MQTT spécifique à l'identifiant du client :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action":["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}"]
      },
      {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:client/client1"]
      }
    ]
}
```

Exemples de stratégies de messages conservés

L’utilisation des messages conservés nécessite des politiques spécifiques. Les messages conservés sont des messages MQTT publiés avec l'indicateur RETAIN défini et stockés par AWS IoT Core. Cette section présente des exemples de politiques qui permettent des utilisations courantes des messages conservés.

Dans cette section :

- Stratégie de connexion et de publication des messages conservés
- Stratégie de connexion et de publication des messages Will conservés
- Stratégie pour répertorier et obtenir les messages conservés

Stratégie de connexion et de publication des messages conservés

Pour qu'un appareil publie des messages conservés, il doit être capable de se connecter, de publier (n'importe quel message MQTT) et de publier des messages MQTT conservés. La stratégie suivante accorde ces autorisations pour la rubrique : `device/sample/configuration` au client **`device1`**. Pour un autre exemple qui accorde l'autorisation de se connecter, consultez the section called “Exemples de stratégies de connexion et de publication”.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/device/sample/configuration"
   ]
  }
 ]
}
```

Stratégie de connexion et de publication des messages Will conservés

Les clients peuvent configurer un message qui AWS IoT Core sera publié lorsqu'ils se déconnecteront de façon inattendue. MQTT appelle un tel message un message Will. Un client doit avoir une condition supplémentaire ajoutée à son autorisation de connexion pour les inclure.

Le document de stratégie suivant accorde à tous les clients l'autorisation de se connecter et de publier un message Will, identifié par sa rubrique, `will`, que AWS IoT Core conservera également.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ],
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/will"
   ]
  }
 ]
}
```

Stratégie pour répertorier et obtenir les messages conservés

Les services et applications peuvent accéder aux messages conservés sans avoir besoin de prendre en charge un client MQTT en appelant `ListRetainedMessages` et `GetRetainedMessage`. Les services et applications qui appellent ces actions doivent être autorisés à l'aide d'une stratégie telle que l'exemple suivant.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:ListRetainedMessages"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ],
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:GetRetainedMessage"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/foo"
   ]
  }
 ]
}
```

Exemples de stratégies de certificat

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante autorise la connexion à l' AWS IoT Core aide d'un ID client correspondant au nom d'un objet et la publication sur une rubrique dont le nom est égal à celui `certificateId` du certificat utilisé par l'appareil pour s'authentifier :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante autorise la connexion AWS IoT Core avec le client IDs, `client1client2`, `client3` et la publication sur une rubrique dont le nom est égal à celui `certificateId` du certificat utilisé par l'appareil pour s'authentifier :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
```

```
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante autorise la connexion à l' AWS IoT Core aide d'un ID client correspondant au nom de l'objet et la publication dans une rubrique dont le nom est égal au `CommonName` champ du sujet du certificat utilisé par l'appareil pour s'authentifier :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
```

```
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

> **Note**
>
> Dans cet exemple, le nom commun de l'objet du certificat est utilisé comme identifiant de rubrique, en supposant que le nom commun de l'objet est unique pour chaque certificat enregistré. Si les certificats sont partagés entre plusieurs appareils, le nom commun de l'objet est le même pour tous les appareils qui partagent ce certificat, ce qui autorise la publication dans la même rubrique à partir de plusieurs appareils (non recommandé).

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante autorise la connexion AWS IoT Core avec le client IDs`client1`,`client2`, `client3` et la publication sur une rubrique dont le nom est égal au `CommonName` champ du sujet du certificat utilisé par l'appareil pour s'authentifier :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

> **Note**
>
> Dans cet exemple, le nom commun de l'objet du certificat est utilisé comme identifiant de rubrique, en supposant que le nom commun de l'objet est unique pour chaque certificat enregistré. Si les certificats sont partagés entre plusieurs appareils, le nom commun de l'objet est le même pour tous les appareils qui partagent ce certificat, ce qui autorise la publication dans la même rubrique à partir de plusieurs appareils (non recommandé).

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante autorise la connexion à l' AWS IoT Core aide d'un ID client correspondant au nom de l'objet et la publication dans une rubrique dont le nom est préfixé `admin/` lorsque le `Subject.CommonName.2` champ du certificat utilisé pour authentifier l'appareil est défini sur : `Administrator`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
```

```
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
                }
            }
        }
    ]
}
```

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante accorde l'autorisation de se connecter au AWS IoT Core client IDs `client1` `client3` et de publier sur une rubrique dont le nom est préfixé `admin/` lorsque le `Subject.CommonName.2` champ du certificat utilisé pour authentifier l'appareil est défini sur : `client2` `Administrator`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
```

```
                }
            }
        }
    ]
}
```

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante permet à un appareil d'utiliser son nom d'objet pour publier sur un sujet spécifique, `admin/` suivi du `ThingName` moment où l'un des `Subject.CommonName` champs du certificat utilisé pour authentifier l'appareil est défini sur : `Administrator`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/
${iot:Connection.Thing.ThingName}"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
                }
            }
        }
    ]
}
```

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante autorise la connexion AWS IoT Core au client IDs `client1` `client3` et la publication dans la rubrique `admin` lorsque l'un des `Subject.CommonName` champs du certificat utilisé pour authentifier l'appareil est défini sur : `client2` `Administrator`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
                }
            }
        }
    ]
}
```

Exemples de stratégies d'objet

La politique suivante permet à un appareil de se connecter si le certificat utilisé pour s'authentifier AWS IoT Core est attaché à l'objet pour lequel la politique est évaluée :

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":["iot:Connect"],
            "Resource":[ "*" ],
            "Condition": {
                "Bool": {
                    "iot:Connection.Thing.IsAttached": ["true"]
                }
            }
        }
    ]
}
```

La stratégie suivante autorise un appareil à publier si le certificat est attaché à un objet avec un type d'objet particulier et celui-ci a un attribut `attributeName` avec une valeur `attributeValue`. Pour plus d'informations sur les variables de stratégie d'objet, consultez Variables de stratégie d'objet.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/device/stats",
      "Condition": {
        "StringEquals": {
          "iot:Connection.Thing.Attributes[attributeName]": "attributeValue",
          "iot:Connection.Thing.ThingTypeName": "Thing_Type_Name"
        },
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
```

```
        }
      }
    }
  ]
}
```

La stratégie suivante permet à un appareil de publier sur une rubrique qui commence par un attribut de l'objet. Si le certificat de périphérique n'est pas associé à l'objet, cette variable ne sera pas résolue et entraînera une erreur d'accès refusé. Pour plus d'informations sur les variables de stratégie d'objet, consultez Variables de stratégie d'objet.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.Attributes[attributeName]}/*"
    }
  ]
}
```

Exemple de stratégie d'emploi de base

Cet exemple montre les déclarations de stratégie requises pour qu'une cible de tâche qui est un appareil unique reçoive une demande de tâche et communique l'état d'exécution de la tâche avec AWS IoT.

Remplacez-le *us-west-2:57EXAMPLE833* par votre Région AWS, deux points (:)) et votre Compte AWS numéro à 12 chiffres, puis remplacez *uniqueThingName* par le nom de l'objet dans AWS IoT lequel la ressource représente le périphérique.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/uniqueThingName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/job/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/
jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/events/jobExecution/
*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/
things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/
jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotjobsdata:DescribeJobExecution",
        "iotjobsdata:GetPendingJobExecutions",
        "iotjobsdata:StartNextPendingJobExecution",
        "iotjobsdata:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName"
      ]
    }
  ]
}
```

Autorisation avec les identités Amazon Cognito

Il existe deux types d'identités Amazon Cognito : non authentifiées et authentifiées. Si votre application prend en charge les identités Amazon Cognito non authentifiées, aucune authentification n'est effectuée, vous ne sauriez donc pas qui est l'utilisateur.

Unauthenticated Identities : Pour les identités Amazon Cognito non authentifiées, vous accordez des autorisations en attachant un rôle IAM à une réserve d'identités non authentifiées. Nous vous recommandons d'accorder uniquement l'accès aux ressources que vous souhaitez rendre accessibles aux utilisateurs inconnus.

> **Important**
>
> Pour les utilisateurs non authentifiés d'Amazon Cognito qui se connectent AWS IoT Coreà, nous vous recommandons de donner accès à des ressources très limitées dans les politiques IAM.

Identités authentifiées : pour les identités Amazon Cognito authentifiées, vous devez spécifier les autorisations à deux endroits :

- Attachez une Politique IAM au réserve d'identités Amazon Cognito authentifié et

- Associez une AWS IoT Core politique à l'identité Amazon Cognito (utilisateur authentifié).

Exemples de stratégie pour les utilisateurs Amazon Cognito non authentifiés et authentifiés se connectant à AWS IoT Core

L'exemple suivant montre les autorisations dans la Politique IAM et IoT d'une identité Amazon Cognito. L'utilisateur authentifié souhaite publier sur un sujet spécifique à l'appareil (par exempledevice/DEVICE_ID/status).

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/Client_ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/device/Device_ID/
status"
            ]
        }
    ]
}
```

L'exemple suivant montre les autorisations dans une Politique IAM d'un rôle non authentifié Amazon Cognito. L'utilisateur non authentifié souhaite publier sur des rubriques non spécifiques à un appareil qui ne nécessitent pas d'authentification.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-
east-1:123456789012:topic/non_device_specific_topic"
            ]
        }
    ]
}
```

GitHub exemples

Les exemples d'applications Web suivants GitHub montrent comment intégrer l'attachement aux politiques des utilisateurs authentifiés dans le processus d'inscription et d'authentification des utilisateurs.

- Application Web MQTT publish/subscribe React utilisant AWS Amplify et Kit SDK des appareils AWS IoT pour JavaScript

- Application Web MQTT publish/subscribe React utilisant AWS AmplifyKit SDK des appareils AWS IoT pour JavaScript, la et une fonction Lambda

Amplify est un ensemble d'outils et de services qui vous aident à créer des applications Web et mobiles qui s'intègrent aux AWS services. Pour plus d'informations sur Amplify, consultez Documentation du Cadre Amplify,.

Les deux exemples effectuent les étapes suivantes.

1. Lorsqu'un utilisateur crée un compte, l'application crée un groupe d'utilisateurs et une identité Amazon Cognito.
2. Lorsqu'un utilisateur s'authentifie, l'application crée et attache une stratégie à l'identité. Cela donne à l'utilisateur des autorisations de publication et d'abonnement.
3. L'utilisateur peut utiliser l'application pour publier et s'abonner à des rubriques MQTT.

Le premier exemple utilise l'opération API `AttachPolicy` directement dans l'opération d'authentification. L'exemple suivant montre comment implémenter cet appel d'API dans une application Web React qui utilise Amplify et le Kit SDK des appareils AWS IoT pour JavaScript.

```
function attachPolicy(id, policyName) {
    var Iot = new AWS.Iot({region: AWSConfiguration.region, apiVersion:
 AWSConfiguration.apiVersion, endpoint: AWSConfiguration.endpoint});
    var params = {policyName: policyName, target: id};

    console.log("Attach IoT Policy: " + policyName + " with cognito identity id: " +
 id);
    Iot.attachPolicy(params, function(err, data) {
         if (err) {
               if (err.code !== 'ResourceAlreadyExistsException') {
                  console.log(err);
               }
          }
         else  {
            console.log("Successfully attached policy with the identity", data);
         }
     });
}
```

Ce code apparaît dans le fichier AuthDisplay.js.

Le deuxième exemple implémente l'opération API `AttachPolicy` dans une fonction Lambda. L'exemple suivant montre comment Lambda utilise cet appel d'API.

```
iot.attachPolicy(params, function(err, data) {
     if (err) {
           if (err.code !== 'ResourceAlreadyExistsException') {
              console.log(err);
              res.json({error: err, url: req.url, body: req.body});
           }
      }
     else  {
        console.log(data);
        res.json({success: 'Create and attach policy call succeed!', url: req.url,
 body: req.body});
     }
 });
```

Ce code apparaît à l'intérieur de la fonction `iot.GetPolicy` dans le fichier app.js.

> **Note**
>
> Lorsque vous appelez la fonction avec les AWS informations d'identification que vous obtenez via les pools Amazon Cognito Identity, l'objet de contexte de votre fonction Lambda contient une valeur pour. `context.cognito_identity_id` Pour plus d'informations, consultez les rubriques suivantes.
>
> - AWS Lambda objet de contexte dans Node.js
> - AWS Lambda objet de contexte en Python
> - AWS Lambda objet de contexte dans Ruby
> - AWS Lambda objet de contexte en Java
> - AWS Lambda objet de contexte dans Go
> - AWS Lambda objet de contexte en C#
> - AWS Lambda objet de contexte dans PowerShell

Autoriser les appels directs vers les AWS services à l'aide d'un fournisseur AWS IoT Core d'informations d'identification

Les appareils peuvent utiliser des certificats X.509 pour se connecter à AWS IoT Core l'aide des protocoles d'authentification mutuelle TLS. AWS Les autres services ne prennent pas en charge l'authentification par certificat, mais ils peuvent être appelés à l'aide AWS d'informations d'identification au format AWS Signature Version 4. L'algorithme Signature Version 4 exige normalement que l'appelant dispose d'un identifiant de clé d'accès et d'une clé d'accès secrète. AWS IoT Core dispose d'un fournisseur d'informations d'identification qui vous permet d'utiliser le certificat X.509 intégré comme identité unique de l'appareil pour AWS authentifier les demandes. Ainsi, vous n'avez plus besoin de stocker un ID de clé d'accès et une clé d'accès secrète sur votre appareil.

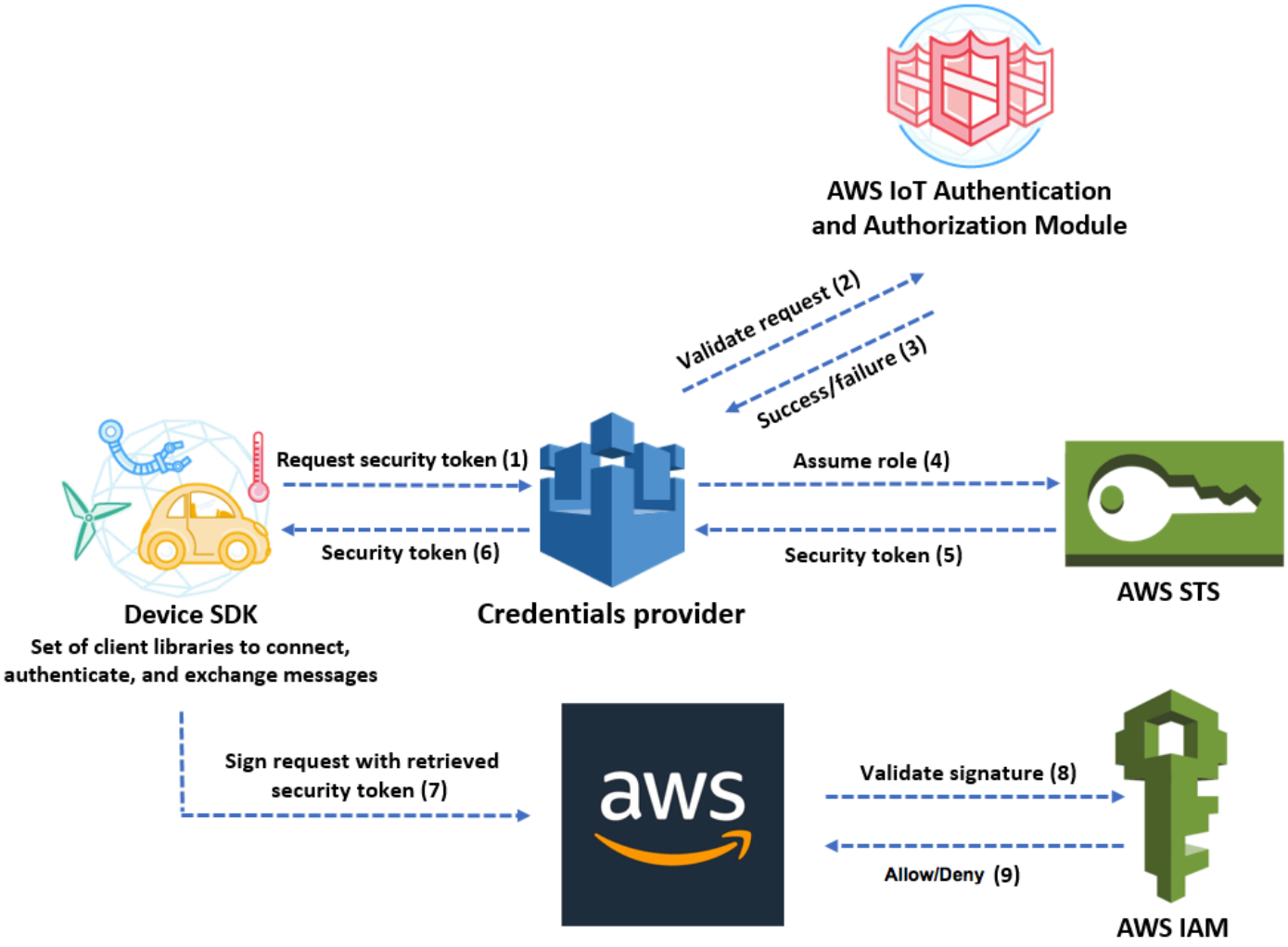
Le fournisseur d'informations d'identification authentifie un mandataire en utilisant un certificat X.509 et émet un jeton de sécurité temporaire à privilèges limités. Le jeton peut être utilisé pour signer et authentifier n'importe quelle AWS demande. Cette méthode d'authentification de vos AWS demandes vous oblige à créer et à configurer un rôle AWS Identity and Access Management (IAM) et à associer les politiques IAM appropriées au rôle afin que le fournisseur d'informations d'identification puisse assumer le rôle en votre nom. Pour plus d'informations sur AWS IoT Core IAM, consultez Gestion des identités et des accès pour AWS IoT.

AWS IoT exige que les appareils envoient l'extension SNI (Server Name Indication) au protocole TLS (Transport Layer Security) et fournissent l'adresse complète du point de terminaison sur le `host_name` terrain. Le champ `host_name` doit contenir le point de terminaison que vous appelez, et il doit être :

- L'`endpointAddress` renvoyée par `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

Les connexions tentées par des appareils sans la valeur `host_name` correcte échoueront.

Le schéma suivant illustre le flux de travail du fournisseur d'informations d'identification.



1. L' AWS IoT Core appareil envoie une demande HTTPS au fournisseur d'informations d'identification pour obtenir un jeton de sécurité. La demande inclut le certificat X.509 de l'appareil pour l'authentification.
2. Le fournisseur d'informations d'identification transmet la demande au module AWS IoT Core d'authentification et d'autorisation pour valider le certificat et vérifier que l'appareil est autorisé à demander le jeton de sécurité.
3. Si le certificat est valide et qu'il est autorisé à demander un jeton de sécurité, le module AWS IoT Core d'authentification et d'autorisation indique la réussite. Dans le cas contraire, il envoie une exception à l'appareil.
4. Une fois que le fournisseur d'informations d'identification a validé le certificat, il appelle AWS Security Token Service (AWS STS) pour endosser le rôle IAM que vous avez créé à son intention.
5. AWS STS renvoie un jeton de sécurité temporaire à privilèges limités au fournisseur d'informations d'identification.

6. Le fournisseur d'informations d'identification renvoie le jeton de sécurité à l'appareil.
7. L'appareil utilise le jeton de sécurité pour signer une AWS demande avec AWS Signature Version 4.
8. Le service demandé invoque IAM à valider la signature et à autoriser la demande par rapport aux stratégies d'accès attachées au rôle IAM que vous avez créé pour le fournisseur d'informations d'identification.
9. Si IAM valide la signature avec succès et autorise la demande, celle-ci aboutit. Sinon, IAM envoie une exception.

La section suivante explique comment utiliser un certificat pour obtenir un jeton de sécurité. Elle est rédigée en partant du principe que vous avez déjà enregistré un appareil et créé, puis activé votre propre certificat pour celui-ci.

Comment utiliser un certificat pour obtenir un jeton de sécurité

1. Configurez le rôle IAM que le fournisseur d'informations d'identification endosse au nom de votre appareil. Attachez la stratégie d'approbation suivante au rôle.

   JSON

   ```
   {
       "Version": "2012-10-17",
       "Statement": {
           "Effect": "Allow",
           "Principal": {"Service": "credentials.iot.amazonaws.com"},
           "Action": "sts:AssumeRole"
       }
   }
   ```

   Pour chaque AWS service que vous souhaitez appeler, associez une politique d'accès au rôle. Le fournisseur d'informations d'identification prend en charge les variables de stratégie suivantes :

   - `credentials-iot:ThingName`
   - `credentials-iot:ThingTypeName`
   - `credentials-iot:AwsCertificateId`

Lorsque l'appareil fournit le nom d'objet dans sa demande adressée à un service AWS , le fournisseur d'informations d'identification ajoute `credentials-iot:ThingName` et `credentials-iot:ThingTypeName` en tant que variables de contexte au jeton de sécurité. Le fournisseur d'informations d'identification fournit `credentials-iot:AwsCertificateId` en tant que variable de contexte, même si l'appareil ne fournit pas le nom d'objet dans la demande. Vous transmettez le nom d'objet comme valeur de l'en-tête de la demande HTTP `x-amzn-iot-thingname`.

Ces trois variables opèrent uniquement pour les stratégies IAM, et non pour les stratégies AWS IoT Core .

2. Vérifiez que l'utilisateur qui effectue l'étape suivante (création d'un alias de rôle) est autorisé à transmettre le rôle nouvellement créé à AWS IoT Core. La politique suivante accorde à la fois `iam:GetRole` des `iam:PassRole` autorisations et des autorisations à un AWS utilisateur. L'autorisation `iam:GetRole` autorise l'utilisateur à obtenir des informations sur le rôle que vous venez de créer. L'`iam:PassRole`autorisation permet à l'utilisateur de transmettre le rôle à un autre AWS service.

   JSON

   ```
   {
       "Version": "2012-10-17",
       "Statement": {
           "Effect": "Allow",
           "Action": [
               "iam:GetRole",
               "iam:PassRole"
           ],
           "Resource": "arn:aws:iam::your Compte AWS id:role/your role name"
       }
   }
   ```

3. Créez un alias de AWS IoT Core rôle. L'appareil qui va passer des appels directs aux AWS services doit savoir quel rôle ARN utiliser pour se connecter AWS IoT Core. Coder en dur l'ARN de rôle n'est pas une bonne solution, car cela vous contraint de mettre à jour l'appareil chaque fois que l'ARN de rôle est modifié. Il vaut mieux utiliser l'API `CreateRoleAlias` pour créer un alias de rôle qui pointe vers l'ARN de rôle. Si l'ARN de rôle est modifié, il vous suffit de mettre

à jour l'alias de rôle. Aucune modification n'est nécessaire sur l'appareil. Cette API accepte les paramètres suivants :

`roleAlias`

Obligatoire. Chaîne arbitraire qui identifie l'alias de rôle. Elle fait office de clé primaire dans le modèle de données d'alias de rôle. Elle contient entre 1 et 128 caractères et doit se composer uniquement de caractères alphanumériques et de symboles =, @ et -. Les caractères alphabétiques majuscules et minuscules sont autorisés. Les noms d'alias de rôle distinguent les majuscules et minuscules.

`roleArn`

Obligatoire. ARN du rôle auquel l'alias de rôle fait référence.

`credentialDurationSeconds`

Facultatif. Durée de validité (en secondes) des informations d'identification. La valeur minimale est de 900 secondes (15 minutes). La valeur maximale est de 43 200 secondes (12 heures). La valeur par défaut est de 3 600 secondes (1 heure).

> **Important**
>
> Le fournisseur AWS IoT Core d'identifiants peut délivrer un identifiant dont la durée de vie maximale est de 43 200 secondes (12 heures). Le fait que les informations d'identification soient valides jusqu'à 12 heures peut contribuer à réduire le nombre d'appels au fournisseur d'informations d'identification en mettant les informations d'identification en cache plus longtemps.
> La valeur `credentialDurationSeconds` doit être inférieure ou égale à la durée maximale de session du rôle IAM auquel l'alias de rôle fait référence. Pour plus d'informations, consultez la section Modification de la durée maximale de session (AWS API) d'un rôle dans le guide de l'utilisateur d' AWS Identity and Access Management.

Pour plus d'informations sur cette API, consultez CreateRoleAlias.

4. Attachez une stratégie au certificat de l'appareil. La stratégie attachée au certificat de l'appareil doit accorder à l'appareil l'autorisation d'assumer le rôle. Pour ce faire, vous devez accorder une

autorisation à l'alias de rôle pour l'action `iot:AssumeRoleWithCertificate`, comme dans l'exemple suivant.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"iot:AssumeRoleWithCertificate",
         "Resource":"arn:aws:iot:your region:your_aws_account_id:rolealias/
your role alias"
      }
   ]
}
```

5. Adressez une demande HTTPS au fournisseur d'informations d'identification pour obtenir un jeton de sécurité. Fournissez les informations suivantes :

   - Certificate : s'agissant d'une demande HTTP avec authentification mutuelle TLS, vous devez fournir le certificat et la clé privée à votre client lorsque vous faites la demande. Utilisez le même certificat et la même clé privée que ceux que vous avez utilisés lors de l'enregistrement de votre certificat AWS IoT Core.

     Pour vous assurer que votre appareil communique avec lui AWS IoT Core (et non avec un service se faisant passer pour lui), consultez Authentification du serveur, suivez les liens pour télécharger les certificats CA appropriés, puis copiez-les sur votre appareil.

   - RoleAlias: nom de l'alias de rôle que vous avez créé pour le fournisseur d'informations d'identification. Les noms d'alias de rôle distinguent les majuscules et minuscules et doivent correspondre à l'alias de rôle créé dans AWS IoT Core.

   - ThingName: nom d'objet que vous avez créé lors de l'enregistrement de votre AWS IoT Core objet. Celui-ci est transmis comme valeur de l'en-tête HTTP `x-amzn-iot-thingname`. Cette valeur n'est requise que si vous utilisez des attributs d'objet comme variables de stratégie dans AWS IoT Core les politiques IAM.

> **Note**
>
> Le nom ThingNameque vous fournissez `x-amzn-iot-thingname` doit correspondre au nom de la ressource d' AWS IoT objet affectée à un certificat. Si cela ne correspond pas, une erreur 403 est renvoyée.

Exécutez la commande suivante dans le AWS CLI pour obtenir le point de terminaison du fournisseur d'informations d'identification pour votre Compte AWS. Pour plus d'informations sur cette API, consultez DescribeEndpoint. Pour les terminaux compatibles FIPS, voir. AWS IoT Core- points de terminaison du fournisseur d'informations d'identification

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

L'objet JSON ci-dessous est un exemple de sortie de la commande describe-endpoint. Il contient le paramètre `endpointAddress` que vous utilisez pour demander un jeton de sécurité.

```
{
    "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your
 region.amazonaws.com"
}
```

Utilisez le point de terminaison pour adresser une demande HTTPS au fournisseur d'informations d'identification pour qu'il renvoie un jeton de sécurité. L'exemple de commande suivant utilise `curl`, mais vous pouvez utiliser n'importe quel client HTTP.

> **Note**
>
> Le nom RoleAlias distingue les majuscules et minuscules et doit correspondre à l'alias de rôle créé dans. AWS IoT

```
curl --cert your certificate --key your private key -H "x-amzn-iot-thingname: your
 thing name" --cacert AmazonRootCA1.pem https://your endpoint /role-aliases/your
 role alias/credentials
```

Cette commande renvoie un objet de jeton de sécurité qui contient les éléments `accessKeyId`, `secretAccessKey`, `sessionToken`, ainsi qu'un délai d'expiration. L'objet JSON ci-dessous est un exemple de sortie de la commande `curl`.

```
   {"credentials":{"accessKeyId":"access key","secretAccessKey":"secret access
key","sessionToken":"session token","expiration":"2018-01-18T09:18:06Z"}}
```

Vous pouvez ensuite utiliser les `sessionToken` valeurs `accessKeyIdsecretAccessKey`, et pour signer les demandes adressées aux AWS services. Pour une end-to-end démonstration, consultez l'article Comment éliminer le besoin d'informations d' AWS identification codées en dur sur les appareils en utilisant le billet de blog du fournisseur AWS IoT d'informations d'identification sur le blog sur la AWS sécurité.

Accès intercompte avec IAM

AWS IoT Core vous permet d'autoriser un directeur à publier ou à s'abonner à un sujet défini dans un sujet qui Compte AWS n'appartient pas au principal. Vous configurez l'accès entre comptes en créant une politique IAM et un rôle IAM, puis en attachant la stratégie au rôle.

Tout d'abord, créez une politique IAM gérée par le client (voir Création de stratégies IAM, comme vous le feriez pour d'autres utilisateurs et certificats de votre Compte AWS.

Pour les appareils enregistrés dans le AWS IoT Core registre, la politique suivante autorise les appareils à se connecter à AWS IoT Core l'aide d'un identifiant client correspondant au nom de l'objet de l'appareil et à publier sur le site `my/topic/thing-name` où se *thing-name* trouve le nom de l'objet de l'appareil :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
```

```
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/my/topic/
${iot:Connection.Thing.ThingName}"],
        }
    ]
}
```

Pour les appareils non enregistrés dans le AWS IoT Core registre, la politique suivante autorise un appareil à utiliser le nom d'objet `client1` enregistré dans le AWS IoT Core registre de votre compte (123456789012) pour se connecter AWS IoT Core et publier sur une rubrique spécifique à l'identifiant client dont le nom est préfixé par : `my/topic/`

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/
${iot:ClientId}"
            ]
        }
    ]
}
```

Suivez ensuite les étapes décrites dans Création d'un rôle pour déléguer des autorisations à un utilisateur IAM. Saisissez l'ID du compte Compte AWS avec lequel vous souhaitez partager l'accès. Puis, dans la dernière étape, attachez la stratégie que vous venez de créer au rôle. Si, ultérieurement, vous avez besoin de modifier l'ID de compte AWS auquel vous accordez l'accès, vous pouvez utiliser le format de stratégie d'approbation suivant à cet effet :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam:us-east-1:567890123456:user/MyUser"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Protection des données dans AWS IoT Core

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans AWS IoT Core. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de

responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec AWS IoT ou d'autres Services AWS utilisateurs de la console, de l'API ou AWS SDKs. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Pour en savoir plus sur la protection des données, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD sur le Blog sur la sécurité d'AWS .

AWS IoT les appareils collectent des données, les manipulent, puis les envoient à un autre service Web. Vous pouvez choisir de stocker certaines données sur votre appareil pendant une courte durée. Il vous incombe d'assurer la protection de ces données au repos. Lorsque votre appareil envoie des données à AWS IoT, il le fait via une connexion TLS, comme indiqué plus loin dans cette section. AWS IoT les appareils peuvent envoyer des données à n'importe quel AWS service. Pour plus d'informations sur la sécurité des données de chaque service, consultez la documentation de ce service. AWS IoT peut être configuré pour écrire des journaux dans les CloudWatch journaux et enregistrer les appels d' AWS IoT API dans AWS CloudTrail. Pour plus d'informations sur la sécurité des données pour ces services, consultez Authentification et contrôle d'accès pour Amazon CloudWatch et Chiffrement des fichiers CloudTrail journaux avec des clés gérées par AWS KMS.

Chiffrement des données dans AWS IoT

Par défaut, toutes les AWS IoT données en transit et au repos sont cryptées. Les données en transit sont chiffrées à l'aide du protocole TLS, tandis que les données au repos sont chiffrées à l'aide de clés AWS détenues. AWS IoT ne prend actuellement pas en charge la gestion par le client AWS KMS keys (clés KMS) à partir du service de gestion des AWS clés (AWS KMS) ; toutefois, Device Advisor et AWS IoT Wireless utilisent uniquement un Clé détenue par AWS pour chiffrer les données des clients.

Sûreté des transports dans AWS IoT Core

TLS (Transport Layer Security) est un protocole cryptographique conçu pour sécuriser les communications sur un réseau informatique. La passerelle pour AWS IoT Core appareils oblige les clients à chiffrer toutes les communications pendant le transport en utilisant le protocole TLS pour les connexions entre les appareils et la passerelle. Le protocole TLS est utilisé pour garantir la confidentialité des protocoles d'application (MQTT, HTTP et WebSocket) pris en charge par. AWS IoT Core La prise en charge de TLS est disponible dans un certain nombre de langages de programmation et de systèmes d'exploitation. Les données AWS qu'elles contiennent sont cryptées par le AWS service spécifique. Pour plus d'informations sur le chiffrement des données sur d'autres AWS services, consultez la documentation de sécurité de ce service.

Table des matières

- Protocoles TLS
- Stratégies de sécurité
- Remarques importantes pour la sécurité des transports en AWS IoT Core
- Sécurité du transport pour les appareils sans fil LoRa WAN

Protocoles TLS

AWS IoT Core prend en charge les versions suivantes du protocole TLS :

- TLS 1.3
- TLS 1.2

Avec AWS IoT Core, vous pouvez configurer les paramètres TLS (pour TLS 1.2 et TLS 1.3) dans les configurations de domaine. Pour de plus amples informations, veuillez consulter ???.

Stratégies de sécurité

Une stratégie de sécurité est une combinaison de protocoles TLS et de leurs chiffrements qui déterminent quels protocoles et chiffrements sont pris en charge lors des négociations TLS entre un client et un serveur. Vous pouvez configurer vos appareils pour utiliser des politiques de sécurité prédéfinies en fonction de vos besoins. Notez que les politiques de sécurité personnalisées AWS IoT Core ne sont pas prises en charge.

Vous pouvez choisir l'une des politiques de sécurité prédéfinies pour vos appareils lorsque vous les connectez à AWS IoT Core. Les noms des politiques de sécurité prédéfinies les plus récentes AWS IoT Core incluent des informations de version basées sur l'année et le mois de leur publication. La stratégie de sécurité prédéfinie par défaut est `IoTSecurityPolicy_TLS13_1_2_2022_10`. Pour définir une politique de sécurité, vous pouvez utiliser la AWS IoT console ou le AWS CLI. Pour de plus amples informations, veuillez consulter ???.

Le tableau suivant décrit les politiques de sécurité prédéfinies les plus récentes que AWS IoT Core prend en charge. Le `IotSecurityPolicy_` a été supprimé des noms de stratégie dans la ligne d'en-tête afin qu'ils correspondent.

Politique de sécurité	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
Ports TCP	443/8443/8883	443/8443/8883	443/8443/8883	443	8443/8883	443	8443/8883
Protocoles TLS							
TLS 1.2		✓	✓	✓	✓	✓	✓

Politique de sécurité	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
TLS 1.3	✓	✓					
Chiffrements TLS							
TLS_AES_128_GCM_SHA256	✓	✓					
TLS_AES_256_GCM_SHA384	✓	✓					
TLS_0_05_CHACHA20_POLY1305_SHA256	✓	✓					
ECDHE-RSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-SHA		✓	✓	✓	✓	✓	✓

Politique de sécurité	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-RSA- -GCM-AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA- -AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-RSA- -SHA AES256		✓	✓	✓	✓	✓	✓
AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
AES128-SHA256		✓	✓	✓		✓	✓
AES128-SHA		✓	✓	✓	✓	✓	✓
AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓
AES256-SHA256		✓	✓	✓	✓	✓	✓
AES256-SHA		✓	✓	✓	✓	✓	✓

Politique de sécurité	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
DHE-RSA--SHA AES256						✓	✓
ECDHE-ECDSA--GCM-AES128 SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-- AES128 SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--SHA AES128		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA--GCM-AES256 SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-- AES256 SHA384		✓	✓	✓	✓	✓	✓

Politique de sécurité	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-ECDSA--SHA AES256		✓	✓	✓	✓	✓	✓

> **Note**
>
> `TLS12_1_0_2016_01`est uniquement disponible dans les pays suivants Régions AWS : ap-east-1, ap-northeast-2, ap-northeast-2, ap-south-1, ap-southeast-2, ca-central-1, cn-north-1, cn-northwest-1, eu-north-1, eu-west-2, eu-west-3, me-south-1, sa-east-1, us-east-1, us-east-2, -1, -2, us-west-1. us-gov-west us-gov-west
> `TLS12_1_0_2015_01`n'est disponible que dans les versions suivantes Régions AWS : ap-northeast-1, ap-southeast-1, eu-central-1, eu-west-1, us-east-1, us-west-2.

Remarques importantes pour la sécurité des transports en AWS IoT Core

Pour les appareils qui se connectent AWS IoT Core via MQTT, le protocole TLS chiffre la connexion entre les appareils et le broker, et AWS IoT Core utilise l'authentification du client TLS pour identifier les appareils. Pour plus d'informations, consultez Authentification client. Pour les appareils qui se connectent AWS IoT Core via HTTP, le protocole TLS chiffre la connexion entre les appareils et le broker, et l'authentification est déléguée à AWS Signature Version 4. Pour plus d'informations, consultez Signature des demandes avec Signature Version 4 dans AWS Référence Générale.

Lorsque vous connectez des appareils à AWS IoT Core, l'envoi de l'extension SNI (Server Name Indication) n'est pas obligatoire mais fortement recommandé. Pour utiliser des fonctionnalités telles que l'enregistrement multi-comptes, les domaines personnalisés, les points de terminaison VPC et les politiques TLS configurées, vous devez utiliser l'extension SNI et fournir l'adresse complète du point de terminaison dans le champ. `host_name` Le champ `host_name` doit contenir le point de terminaison que vous invoquez. Ce point de terminaison doit être l'un des éléments suivants :

- L'adresse `endpointAddress` renvoyée par `aws iot describe-endpoint --endpoint-type iot:Data-ATS`

- L'adresse domainName renvoyée par `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

Les connexions tentées par des appareils dont la `host_name` valeur est incorrecte ou non valide échoueront. AWS IoT Core enregistrera les échecs CloudWatch pour le type d'authentification de l'authentification personnalisée.

AWS IoT Core ne prend pas en charge l'extension SessionTicket TLS.

Sécurité du transport pour les appareils sans fil LoRa WAN

LoRaLes appareils WAN suivent les pratiques de sécurité décrites dans LoRaWAN™ SECURITY : A White Paper for the LoRa Alliance™ par Gemalto, Actility et Semtech.

Pour plus d'informations sur la sécurité du transport avec les périphériques LoRa WAN, consultez la section Données LoRa WAN et sécurité du transport.

Chiffrement des données dans AWS IoT

La protection des données fait référence à la protection des données en transit (lors de leur trajet aller-retour AWS IoT) et au repos (lorsqu'elles sont stockées sur des appareils ou par d'autres AWS services). Toutes les données envoyées AWS IoT sont envoyées via une connexion TLS à l'aide de MQTT, HTTPS et des WebSocket protocoles, ce qui les rend sécurisées par défaut pendant le transit. AWS IoT les appareils collectent des données puis les envoient à d'autres AWS services pour un traitement ultérieur. Pour plus d'informations sur le chiffrement des données dans d'autres services AWS , consultez la documentation relative à la sécurité du service concerné.

FreeRTOS fournit une bibliothèque PKCS#11 qui isole le stockage des clés, en accédant aux objets cryptographiques et en gérant les sessions. Il vous incombe d'utiliser cette bibliothèque pour chiffrer les données au repos sur vos appareils. Pour plus d'informations, consultez la bibliothèque FreeRTOS Public Key Cryptography Standard (PKCS) #11.

Device Advisor

Chiffrement en transit

Les données envoyées vers et depuis Device Advisor sont cryptées pendant leur transit. Toutes les données envoyées vers et depuis le service lors de l'utilisation du Device Advisor APIs sont cryptées

à l'aide de la version 4 de Signature. Pour plus d'informations sur la façon dont les demandes AWS d'API sont signées, consultez la section Signature des demandes AWS d'API. Toutes les données envoyées depuis vos appareils de test vers votre point de terminaison de test Device Advisor sont envoyées via une connexion TLS afin qu'elles soient sécurisées par défaut pendant le transit.

Gestion des clés dans AWS IoT

Toutes les connexions AWS IoT sont effectuées à l'aide du protocole TLS, de sorte qu'aucune clé de chiffrement côté client n'est nécessaire pour la connexion TLS initiale.

Les appareils doivent s'authentifier à l'aide d'un certificat X.509 ou d'une identité Amazon Cognito. Vous pouvez avoir AWS IoT généré un certificat pour vous, auquel cas il générera une paire de public/private clés. Si vous utilisez la AWS IoT console, vous serez invité à télécharger le certificat et les clés. Si vous utilisez la commande `create-keys-and-certificate` de l'interface de ligne de commande, le certificat et les clés sont renvoyés par cette commande. Il vous incombe de copier le certificat et la clé privée sur votre appareil et de veiller à leur sécurité.

AWS IoT ne prend actuellement pas en charge la gestion par le client AWS KMS keys (clés KMS) depuis AWS Key Management Service (AWS KMS) ; toutefois, Device Advisor et AWS IoT Wireless utilisent uniquement an Clé détenue par AWS pour chiffrer les données des clients.

Device Advisor

Toutes les données envoyées à Device Advisor lors de l'utilisation du AWS APIs sont cryptées au repos. Device Advisor crypte toutes vos données au repos à l'aide de clés KMS stockées et gérées dans AWS Key Management Service. Device Advisor chiffre vos données à l'aide Clés détenues par AWS de. Pour plus d'informations sur Clés détenues par AWS, voir Clés détenues par AWS.

Gestion des identités et des accès pour AWS IoT

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser AWS IoT les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé
- Authentification avec des identités IAM

- Gestion des accès à l’aide de politiques
- Comment AWS IoT fonctionne avec IAM
- AWS IoT exemples de politiques basées sur l'identité
- AWS politiques gérées pour AWS IoT
- Résolution des problèmes AWS IoT d'identité et d'accès

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez. AWS IoT

Utilisateur du service : si vous utilisez le AWS IoT service pour effectuer votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de nouvelles AWS IoT fonctionnalités pour effectuer votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans AWS IoT, consultez Résolution des problèmes AWS IoT d'identité et d'accès.

Administrateur du service — Si vous êtes responsable des AWS IoT ressources de votre entreprise, vous avez probablement un accès complet à AWS IoT. C'est à vous de déterminer les AWS IoT fonctionnalités et les ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d’IAM. Pour en savoir plus sur la manière dont votre entreprise peut utiliser IAM avec AWS IoT, voirComment AWS IoT fonctionne avec IAM.

Administrateur IAM – Si vous êtes un administrateur IAM, vous souhaiterez peut-être en savoir plus sur la façon d’écrire des politiques pour gérer l’accès à AWS IoT. Pour consulter des exemples de politiques AWS IoT basées sur l'identité que vous pouvez utiliser dans IAM, consultez. AWS IoT exemples de politiques basées sur l'identité

Authentification avec des identités IAM

Dans les AWS IoT identités, il peut s'agir de certificats d'appareils (X.509), d'identités Amazon Cognito ou d'utilisateurs ou de groupes IAM. Cette rubrique traite uniquement des identités IAM.

Pour plus d'informations sur les autres identités prises AWS IoT en charge, consultezAuthentification client.

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez AWS Signature Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour plus d'informations, consultez Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Authentification multifactorielle AWS dans IAM dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas

utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez Tâches nécessitant les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme telles que des mots de passe et des clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons d'effectuer une rotation des clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Pour assumer temporairement un rôle IAM dans le AWS Management Console, vous pouvez passer d'un rôle d'utilisateur à un rôle IAM (console). Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.
  - Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM.

Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

- Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés au service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des demandes AWS CLI d' AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et AWS WAF Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les Organizations et consultez SCPs les politiques de contrôle des services dans le Guide de AWS Organizations l'utilisateur.
- Politiques de contrôle des ressources (RCPs) : RCPs politiques JSON que vous pouvez utiliser pour définir le maximum d'autorisations disponibles pour les ressources de vos comptes sans mettre à jour les politiques IAM associées à chaque ressource que vous possédez. Le RCP limite les autorisations pour les ressources des comptes membres et peut avoir un impact sur les

autorisations effectives pour les identités, y compris Utilisateur racine d'un compte AWS, qu'elles appartiennent ou non à votre organisation. Pour plus d'informations sur les Organizations RCPs, y compris une liste de ces Services AWS supports RCPs, consultez la section Resource control policies (RCPs) dans le guide de AWS Organizations l'utilisateur.

- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment AWS IoT fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à AWS IoT, vous devez comprendre quelles fonctionnalités IAM sont disponibles avec. AWS IoT Pour obtenir une vue d'ensemble de la façon dont AWS IoT les autres AWS services fonctionnent avec IAM, consultez la section AWS Services qui fonctionnent avec IAM dans le Guide de l'utilisateur d'IAM.

Rubriques

- AWS IoT Politiques basées sur l'identité
- AWS IoT Politiques basées sur les ressources
- Autorisation basée sur les balises AWS IoT
- AWS IoT Rôles IAM

AWS IoT Politiques basées sur l'identité

Avec les politiques basées sur l'identité IAM, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou

refusées. AWS IoT prend en charge des actions, ressources et clés de condition spécifiques. Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Le tableau suivant répertorie les actions IAM IoT, l' AWS IoT API associée et la ressource manipulée par l'action.

Actions de politique	AWS IoT API	Ressources
IoT : AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** Le compte Compte AWS spécifié dans l'ARN doit être le compte vers lequel le certificat est transféré.
IoT : AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*

Actions de politique	AWS IoT API	Ressources
		arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : AssociateTargetsWithJob	AssociateTargetsWithJob	none
IoT : AttachPolicy	AttachPolicy	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* or arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : AttachPrincipalPolicy	AttachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : AttachSecurityProfile	AttachSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT : AttachThingPrincipal	AttachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : CancelCertificateTransfer	CancelCertificateTransfer	arn:aws:iot: *region*:*account-id* :cert/*cert-id* **Note** Le compte Compte AWS spécifié dans l'ARN doit être le compte vers lequel le certificat est transféré.
IoT : CancelJob	CancelJob	arn:aws:iot: *region*:*account-id* :job/*job-id*

Actions de politique	AWS IoT API	Ressources
IoT : CancelJob Execution	CancelJob Execution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : ClearDefaultAuthorizer	ClearDefaultAuthorizer	Aucun
IoT : CreateAuthorizer	CreateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT : CreateCertificateFromCsr	CreateCertificateFromCsr	*
IoT : CreateDimension	CreateDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT : CreateJob	CreateJob	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT : CreateJobTemplate	CreateJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT : CreateKeysAndCertificate	CreateKeysAndCertificate	*

Actions de politique	AWS IoT API	Ressources
IoT : CreatePolicy	CreatePolicy	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name*
IoT : CreatePolicyVersion	CreatePolicyVersion	`arn:aws:iot:` *region*`:`*account-id* `:policy/`*policy-name* **Note** Il doit s'agir d'une AWS IoT politique et non d'une politique IAM.
IoT : CreateRoleAlias	CreateRoleAlias	(paramètre : roleAlias) `arn:aws:iot:` *region*`:`*account-id* `:rolealias/` *role-alias-name*
IoT : CreateSecurityProfile	CreateSecurityProfile	`arn:aws:iot:` *region*`:`*account-id* `:securityprofile/` *security-profile-name* `arn:aws:iot:` *region*`:`*account-id* `:dimension/` *dimension-name*
IoT : CreateThing	CreateThing	`arn:aws:iot:` *region*`:`*account-id* `:thing/`*thing-name*
IoT : CreateThingGroup	CreateThingGroup	`arn:aws:iot:` *region*`:`*account-id* `:thinggroup/` *thing-group-name* pour le groupe en cours de création et pour le groupe parent, si utilisé
IoT : CreateThingType	CreateThingType	`arn:aws:iot:` *region*`:`*account-id* `:thingtype/` *thing-type-name*
IoT : CreateTopicRule	CreateTopicRule	`arn:aws:iot:` *region*`:`*account-id* `:rule/`*rule-name*

Actions de politique	AWS IoT API	Ressources
IoT : DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-name*
IoT : Supprimer CACertificate	Supprimer CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT : DeleteCertificate	DeleteCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : DeleteDimension	DeleteDimension	arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT : DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT : DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-template-id*
IoT : DeleteJobExecution	DeleteJobExecution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : DeleteRegistrationCode	DeleteRegistrationCode	*
IoT : DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*

Actions de politique	AWS IoT API	Ressources
IoT : DeleteSecurityProfile	DeleteSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT : DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : DeleteThingGroup	DeleteThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : DeleteThingType	DeleteThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT : DeleteTopicRule	DeleteTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : DeleteV2LoggingLevel	Supprimer la version 2 LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : DeprecateThingType	DeprecateThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT : DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name* (paramètre : authorizerName) none
IoT : décrire CACertificate	Décrivez CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT : DescribeCertificate	DescribeCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Actions de politique	AWS IoT API	Ressources
IoT : DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Aucun
IoT : DescribeEndpoint	DescribeEndpoint	*
IoT : DescribeEventConfigurations	DescribeEventConfigurations	none
IoT : DescribeIndex	DescribeIndex	arn:aws:iot: *region*:*account-id* :index/*index-name*
IoT : DescribeJob	DescribeJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT : DescribeJobExecution	DescribeJobExecution	Aucun
IoT : DescribeJobTemplate	DescribeJobTemplate	arn:aws:iot: *region*:*account-id* :job/*job-template-id*
IoT : DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT : DescribeThing	DescribeThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : DescribeThingGroup	DescribeThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : DescribeThingRegistrationTask	DescribeThingRegistrationTask	Aucun

Actions de politique	AWS IoT API	Ressources
IoT : DescribeThingType	DescribeThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*
IoT : DetachPolicy	DetachPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id* or arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : DetachSecurityProfile	DetachSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT : DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : DisableTopicRule	DisableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : EnableTopicRule	EnableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : GetIndexingConfiguration	GetIndexingConfiguration	Aucun
IoT : GetJobDocument	GetJobDocument	arn:aws:iot: *region*:*account-id* :job/*job-id*

Actions de politique	AWS IoT API	Ressources
IoT : GetLoggingOptions	GetLoggingOptions	*
IoT : GetPolicy	GetPolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : GetPolicyVersion	GetPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : GetRegistrationCode	GetRegistrationCode	*
IoT : GetTopicRule	GetTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : ListAttachedPolicies	ListAttachedPolicies	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* or arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : ListAuthorizers	ListAuthorizers	Aucun
IoT : liste CACertificates	Liste CACertificates	*
IoT : ListCertificates	ListCertificates	*
Lieu : ListCertificatesBy CA	ListCertificatesByCA	*
IoT : ListIndices	ListIndices	Aucun

Actions de politique	AWS IoT API	Ressources
IoT : ListJobExecutionsForJob	ListJobExecutionsForJob	Aucun
IoT : ListJobExecutionsForThing	ListJobExecutionsForThing	Aucun
IoT : ListJobs	ListJobs	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* si le thingGroupName paramètre est utilisé
IoT : ListJobTemplates	ListJobs	Aucun
IoT : ListOutgoingCertificates	ListOutgoingCertificates	*
IoT : ListPolicies	ListPolicies	*
IoT : ListPolicyPrincipals	ListPolicyPrincipals	*
IoT : ListPolicyVersions	ListPolicyVersions	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT : ListPrincipalPolicies	ListPrincipalPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : ListPrincipalThings	ListPrincipalThings	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : ListRoleAliases	ListRoleAliases	Aucun
IoT : ListTargetsForPolicy	ListTargetsForPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*

Actions de politique	AWS IoT API	Ressources
IoT : ListThing Groups	ListThingGroups	Aucun
IoT : ListThing GroupsForThing	ListThing GroupsForThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : ListThing Principals	ListThing Principals	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : ListThing Registrat ionTaskReports	ListThing Registrat ionTaskReports	Aucun
IoT : ListThing Registrat ionTasks	ListThing Registrat ionTasks	Aucun
IoT : ListThing Types	ListThingTypes	*
IoT : ListThings	ListThings	*
IoT : ListThing sInThingGroup	ListThing sInThingGroup	arn:aws:iot: *region*:*account-id* :thinggro up/ *thing-group-name*
IoT : ListTopic Rules	ListTopicRules	*
IoT : Liste V2 LoggingLevels	Liste V2 LoggingLevels	Aucun
IoT : S'inscrire CACertificate	S'inscrire CACertificate	*
IoT : RegisterC ertificate	RegisterCertificat e	*

Actions de politique	AWS IoT API	Ressources
IoT : RegisterThing	RegisterThing	Aucun
IoT : RejectCertificateTransfer	RejectCertificateTransfer	arn:aws:iot: *region*:*account-id* :cert/*certificate-id*
IoT : RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : ReplaceTopicRule	ReplaceTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : SearchIndex	SearchIndex	arn:aws:iot: *region*:*account-id* :index/*index-id*
IoT : SetDefaultAuthorizer	SetDefaultAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT : SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : SetLoggingOptions	SetLoggingOptions	arn:aws:iot: *region*:*account-id* :role/*role-name*
IoT : SetV2LoggingLevel	Set V2LoggingLevel	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : SetV2LoggingOptions	Set V2LoggingOptions	arn:aws:iot: *region*:*account-id* :role/*role-name*
IoT : StartThingRegistrationTask	StartThingRegistrationTask	Aucun

Actions de politique	AWS IoT API	Ressources
IoT : StopThing RegistrationTask	StopThing RegistrationTask	Aucun
IoT : TestAutho rization	TestAutho rization	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : TestInvok eAuthorizer	TestInvok eAuthorizer	Aucun
IoT : TransferC ertificate	TransferCertificat e	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : UpdateAut horizer	UpdateAut horizer	arn:aws:iot: *region*:*account-id* :authoriz erfunction/ *authorizer-function-name*
IoT : mise à jour CACertificate	Mettre à jour CACertificate	arn:aws:iot: *region*:*account-i d* :cacert/*cert-id*
IoT : UpdateCer tificate	UpdateCer tificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : UpdateDim ension	UpdateDim ension	arn:aws:iot: *region*:*account-id* :dimensio n/ *dimension-name*
IoT : UpdateEve ntConfigurations	UpdateEve ntConfigurations	Aucun
IoT : UpdateInd exingConf iguration	UpdateInd exingConf iguration	Aucun
IoT : UpdateRol eAlias	UpdateRoleAlias	arn:aws:iot: *region*:*account-id* :rolealia s/ *role-alias-name*

Actions de politique	AWS IoT API	Ressources
IoT : UpdateSecurityProfile	UpdateSecurityProfile	arn:aws:iot: *region*:*account-id* :securityprofile/ *security-profile-name* arn:aws:iot: *region*:*account-id* :dimension/ *dimension-name*
IoT : UpdateThing	UpdateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : UpdateThingGroup	UpdateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : UpdateThingGroupsForThing	UpdateThingGroupsForThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name* arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*

Les actions de politique en AWS IoT cours utilisent le préfixe suivant avant l'action :`iot:`. Par exemple, pour autoriser une personne à répertorier tous les objets IoT enregistrés dans Compte AWS l'`ListThings`API, vous devez inclure l'`iot:ListThings`action dans sa politique. Les déclarations de politique doivent inclure un `NotAction` élément `Action` ou. AWS IoT définit son propre ensemble d'actions décrivant les tâches que vous pouvez effectuer avec ce service.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": [
      "ec2:action1",
      "ec2:action2"
```

Vous pouvez aussi spécifier plusieurs actions à l’aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l’action suivante :

```
"Action": "iot:Describe*"
```

Pour consulter la liste des AWS IoT actions, reportez-vous à la section Actions définies par AWS IoT dans le guide de l'utilisateur IAM.

Actions du Device Advisor

Le tableau suivant répertorie les actions IAM IoT Device Advisor, l'API Device Advisor AWS IoT associée et la ressource manipulée par l'action.

Actions de politique	AWS IoT API	Ressources
conseiller pour appareils IoT : CreateSuiteDefinition	CreateSuiteDefinition	Aucun
conseiller pour appareils IoT : DeleteSuiteDefinition	DeleteSuiteDefinition	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`*` :suitedefinition/` *`suite-definition-id`*
conseiller pour appareils IoT : GetSuiteDefinition	GetSuiteDefinition	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`*` :suitedefinition/` *`suite-definition-id`*
conseiller pour appareils IoT : GetSuiteRun	GetSuiteRun	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`*` :suitedefinition/` *`suite-run-id`*
conseiller pour appareils IoT : GetSuiteRunReport	GetSuiteRunReport	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`*` :suiterun/` *`suite-definition-id`*` /`*`suite-run-id`*
conseiller pour appareils IoT : ListSuiteDefinitions	ListSuiteDefinitions	Aucun

Actions de politique	AWS IoT API	Ressources
conseiller pour appareils IoT : ListSuiteRuns	ListSuiteRuns	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
conseiller pour appareils IoT : ListTagsForResource	ListTagsForResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
conseiller pour appareils IoT : StartSuiteRun	StartSuiteRun	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*
conseiller pour appareils IoT : TagResource	TagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
conseiller pour appareils IoT : UntagResource	UntagResource	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id* arn:aws:iotdeviceadvisor: *region*:*account-id* :suiterun/suite-definition-id/ *suite-run-id*
conseiller pour appareils IoT : UpdateSuiteDefinition	UpdateSuiteDefinition	arn:aws:iotdeviceadvisor: *region*:*account-id* :suitedefinition/ *suite-definition-id*

Actions de politique	AWS IoT API	Ressources
conseiller pour appareils IoT : StopSuiteRun	StopSuiteRun	`arn:aws:iotdeviceadvisor:` *`region`*`:`*`account-id`* `:suiterun/suite-definition-id/` *`suite-run-id`*

Les actions de politique dans AWS IoT Device Advisor utilisent le préfixe suivant avant l'action : `iotdeviceadvisor:` Par exemple, pour autoriser quelqu'un à répertorier toutes les définitions de suites enregistrées dans son Compte AWS ListSuiteDefinitions API, vous devez inclure l'`iotdeviceadvisor:ListSuiteDefinitions`action dans sa politique.

Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

AWS IoT ressources

Actions de politique	AWS IoT API	Ressources
IoT : AcceptCertificateTransfer	AcceptCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*

Actions de politique	AWS IoT API	Ressources
		Note Le compte Compte AWS spécifié dans l'ARN doit être le compte vers lequel le certificat est transféré.
IoT : AddThingToThingGroup	AddThingToThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT : AssociateTargetsWithJob	AssociateTargetsWithJob	Aucun
IoT : AttachPolicy	AttachPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* or `arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : AttachPrincipalPolicy	AttachPrincipalPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : AttachThingPrincipal	AttachThingPrincipal	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*

Actions de politique	AWS IoT API	Ressources
IoT : CancelCertificateTransfer	CancelCertificateTransfer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`* **Note** Le compte Compte AWS spécifié dans l'ARN doit être le compte vers lequel le certificat est transféré.
IoT : CancelJob	CancelJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`*
IoT : CancelJobExecution	CancelJobExecution	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT : ClearDefaultAuthorizer	ClearDefaultAuthorizer	Aucun
IoT : CreateAuthorizer	CreateAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`*
IoT : CreateCertificateFromCsr	CreateCertificateFromCsr	*
IoT : CreateJob	CreateJob	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:jobtemplate/` *`job-template-id`*

Actions de politique	AWS IoT API	Ressources
IoT : CreateJobTemplate	CreateJobTemplate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:job/`*`job-id`* `arn:aws:iot:` *`region`*`:`*`account-id`* `:jobtemplate/` *`job-template-id`*
IoT : CreateKeysAndCertificate	CreateKeysAndCertificate	*
IoT : CreatePolicy	CreatePolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
CreatePolicyVersion	IoT : CreatePolicyVersion	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`* **Note** Il doit s'agir d'une AWS IoT politique et non d'une politique IAM.
IoT : CreateRoleAlias	CreateRoleAlias	(paramètre : roleAlias) `arn:aws:iot:` *`region`*`:`*`account-id`* `:rolealias/` *`role-alias-name`*
IoT : CreateThing	CreateThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*
IoT : CreateThingGroup	CreateThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* pour le groupe en cours de création et pour le groupe parent, si utilisé
IoT : CreateThingType	CreateThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*

Actions de politique	AWS IoT API	Ressources
IoT : CreateTopicRule	CreateTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-name*
IoT : SupprimerCACertificate	SupprimerCACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT : DeleteCertificate	DeleteCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : DeleteJob	DeleteJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT : DeleteJobExecution	DeleteJobExecution	arn:aws:iot: *region*:*account-id* :job/*job-id* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT : DeletePolicy	DeletePolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : DeleteRegistrationCode	DeleteRegistrationCode	*
IoT : DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT : DeleteThing	DeleteThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*

Actions de politique	AWS IoT API	Ressources
IoT : DeleteThingGroup	DeleteThingGroup	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT : DeleteThingType	DeleteThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*
IoT : DeleteTopicRule	DeleteTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT : DeleteV2LoggingLevel	Supprimer la version 2 LoggingLevel	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`*
IoT : DeprecateThingType	DeprecateThingType	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thingtype/` *`thing-type-name`*
IoT : DescribeAuthorizer	DescribeAuthorizer	`arn:aws:iot:` *`region`*`:`*`account-id`* `:authorizer/` *`authorizer-function-name`* (paramètre : authorizerName) none
IoT : décrire CACertificate	Décrivez CACertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cacert/`*`cert-id`*
IoT : DescribeCertificate	DescribeCertificate	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	Aucun
IoT : DescribeEndpoint	DescribeEndpoint	*

Actions de politique	AWS IoT API	Ressources
IoT : DescribeEventConfigurations	DescribeEventConfigurations	none
IoT : DescribeIndex	DescribeIndex	arn:aws:iot: *region*:*account-id* :index/*index-name*
IoT : DescribeJob	DescribeJob	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT : DescribeJobExecution	DescribeJobExecution	Aucun
IoT : DescribeJobTemplate	DescribeJobTemplate	arn:aws:iot: *region*:*account-id* :jobtemplate/ *job-template-id*
IoT : DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT : DescribeThing	DescribeThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : DescribeThingGroup	DescribeThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : DescribeThingRegistrationTask	DescribeThingRegistrationTask	Aucun
IoT : DescribeThingType	DescribeThingType	arn:aws:iot: *region*:*account-id* :thingtype/ *thing-type-name*

Actions de politique	AWS IoT API	Ressources
IoT : DetachPolicy	DetachPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id* or arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : DisableTopicRule	DisableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : EnableTopicRule	EnableTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : GetIndexingConfiguration	GetIndexingConfiguration	Aucun
IoT : GetJobDocument	GetJobDocument	arn:aws:iot: *region*:*account-id* :job/*job-id*
IoT : GetLoggingOptions	GetLoggingOptions	*
IoT : GetPolicy	GetPolicy	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : GetPolicyVersion	GetPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*

Actions de politique	AWS IoT API	Ressources
IoT : GetRegistrationCode	GetRegistrationCode	*
IoT : GetTopicRule	GetTopicRule	`arn:aws:iot:` *`region`*`:`*`account-id`* `:rule/`*`rule-name`*
IoT : ListAttachedPolicies	ListAttachedPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* or `arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : ListAuthorizers	ListAuthorizers	Aucun
IoT : liste CACertificates	Liste CACertificates	*
IoT : ListCertificates	ListCertificates	*
Lieu : ListCertificatesBy CA	ListCertificatesByCA	*
IoT : ListIndices	ListIndices	Aucun
IoT : ListJobExecutionsForJob	ListJobExecutionsForJob	Aucun
IoT : ListJobExecutionsForThing	ListJobExecutionsForThing	Aucun

Actions de politique	AWS IoT API	Ressources
IoT : ListJobs	ListJobs	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thinggroup/` *`thing-group-name`* si le thingGroupName paramètre est utilisé
IoT : ListJobTemplates	ListJobTemplates	Aucun
IoT : ListOutgoingCertificates	ListOutgoingCertificates	*
IoT : ListPolicies	ListPolicies	*
IoT : ListPolicyPrincipals	ListPolicyPrincipals	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT : ListPolicyVersions	ListPolicyVersions	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT : ListPrincipalPolicies	ListPrincipalPolicies	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : ListPrincipalThings	ListPrincipalThings	`arn:aws:iot:` *`region`*`:`*`account-id`* `:cert/`*`cert-id`*
IoT : ListRoleAliases	ListRoleAliases	Aucun
IoT : ListTargetsForPolicy	ListTargetsForPolicy	`arn:aws:iot:` *`region`*`:`*`account-id`* `:policy/`*`policy-name`*
IoT : ListThingGroups	ListThingGroups	Aucun
IoT : ListThingGroupsForThing	ListThingGroupsForThing	`arn:aws:iot:` *`region`*`:`*`account-id`* `:thing/`*`thing-name`*

Actions de politique	AWS IoT API	Ressources
IoT : ListThing Principals	ListThing Principals	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : ListThing RegistrationTaskReports	ListThing RegistrationTaskReports	Aucun
IoT : ListThing RegistrationTasks	ListThing RegistrationTasks	Aucun
IoT : ListThing Types	ListThingTypes	*
IoT : ListThings	ListThings	*
IoT : ListThingsInThingGroup	ListThingsInThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : ListTopic Rules	ListTopicRules	*
IoT : Liste V2 LoggingLevels	Liste V2 LoggingLevels	Aucun
IoT : S'inscrire CACertificate	S'inscrire CACertificate	*
IoT : RegisterCertificate	RegisterCertificate	*
IoT : RegisterThing	RegisterThing	Aucun
IoT : RejectCertificateTransfer	RejectCertificateTransfer	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Actions de politique	AWS IoT API	Ressources
IoT : RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name* arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : ReplaceTopicRule	ReplaceTopicRule	arn:aws:iot: *region*:*account-id* :rule/*rule-name*
IoT : SearchIndex	SearchIndex	arn:aws:iot: *region*:*account-id* :index/*index-id*
IoT : SetDefaultAuthorizer	SetDefaultAuthorizer	arn:aws:iot: *region*:*account-id* :authorizer/ *authorizer-function-name*
IoT : SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: *region*:*account-id* :policy/*policy-name*
IoT : SetLoggingOptions	SetLoggingOptions	*
IoT : SetV2LoggingLevel	Set V2LoggingLevel	*
IoT : SetV2LoggingOptions	Set V2LoggingOptions	*
IoT : StartThingRegistrationTask	StartThingRegistrationTask	Aucun
IoT : StopThingRegistrationTask	StopThingRegistrationTask	Aucun
IoT : TestAuthorization	TestAuthorization	arn:aws:iot: *region*:*account-id* :cert/*cert-id*

Actions de politique	AWS IoT API	Ressources
IoT : TestInvokeAuthorizer	TestInvokeAuthorizer	Aucun
IoT : TransferCertificate	TransferCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: *region*:*account-id* :authorizerfunction/ *authorizer-function-name*
IoT : mise à jour CACertificate	Mettre à jour CACertificate	arn:aws:iot: *region*:*account-id* :cacert/*cert-id*
IoT : UpdateCertificate	UpdateCertificate	arn:aws:iot: *region*:*account-id* :cert/*cert-id*
IoT : UpdateEventConfigurations	UpdateEventConfigurations	Aucun
IoT : UpdateIndexingConfiguration	UpdateIndexingConfiguration	Aucun
IoT : UpdateRoleAlias	UpdateRoleAlias	arn:aws:iot: *region*:*account-id* :rolealias/ *role-alias-name*
IoT : UpdateThing	UpdateThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*
IoT : UpdateThingGroup	UpdateThingGroup	arn:aws:iot: *region*:*account-id* :thinggroup/ *thing-group-name*
IoT : UpdateThingGroupsForThing	UpdateThingGroupsForThing	arn:aws:iot: *region*:*account-id* :thing/*thing-name*

Pour plus d'informations sur le format de ARNs, consultez Amazon Resource Names (ARNs) et AWS Service Namespaces.

Certaines AWS IoT actions, telles que celles relatives à la création de ressources, ne peuvent pas être effectuées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (*).

```
"Resource": "*"
```

Pour consulter la liste des types de AWS IoT ressources et leurs caractéristiques ARNs, reportez-vous à la section Ressources définies par AWS IoT dans le guide de l'utilisateur IAM. Pour savoir grâce à quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par AWS IoT.

ressources Device Advisor

Pour définir des restrictions au niveau des ressources pour les politiques IAM de AWS IoT Device Advisor, utilisez les formats ARN des ressources suivants pour les définitions et les exécutions de suites.

Format ARN de la ressource de définition de suite

```
arn:aws:iotdeviceadvisor:region:account-id:suitedefinition/suite-definition-id
```

Format ARN de la ressource d'exécution de la suite

```
arn:aws:iotdeviceadvisor:region:account-id:suiterun/suite-definition-id/suite-run-id
```

Clés de condition

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et identifications dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

AWS IoT définit son propre ensemble de clés de condition et prend également en charge l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, consultez la section Clés contextuelles de condition AWS globale dans le guide de l'utilisateur IAM.

AWS IoT clés de condition

AWS IoT clés de condition	Description	Type
`aws:RequestTag/${tag-key}`	Clé de balise présente dans la demande envoyée par l'utilisateur à AWS IoT.	Chaîne
`aws:ResourceTag/${tag-key}`	Le composant clé d'une balise attachée à une AWS IoT ressource.	Chaîne
`aws:TagKeys`	Liste de tous les noms de clés de balise associés à	Chaîne

AWS IoT clés de condition	Description	Type
	la ressource de la demande.	

Pour consulter la liste des clés de AWS IoT condition, reportez-vous à la section Clés de AWS IoT condition du guide de l'utilisateur IAM. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, voir Actions définies par AWS IoT.

Exemples

Pour consulter des exemples de politiques AWS IoT basées sur l'identité, consultez. AWS IoT exemples de politiques basées sur l'identité

AWS IoT Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON qui spécifient les actions qu'un principal spécifié peut effectuer sur la AWS IoT ressource et dans quelles conditions.

AWS IoT ne prend pas en charge les politiques basées sur les ressources IAM. Il soutient toutefois les politiques AWS IoT basées sur les ressources. Pour de plus amples informations, veuillez consulter AWS IoT Core politiques.

Autorisation basée sur les balises AWS IoT

Vous pouvez associer des balises aux AWS IoT ressources ou transmettre des balises dans une demande à AWS IoT. Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `iot:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou `aws:TagKeys`. Pour de plus amples informations, veuillez consulter Utilisation des balises avec des politiques IAM. Pour plus d'informations sur le balisage AWS IoT des ressources, consultezMarquer vos ressources AWS IoT.

Pour afficher un exemple de stratégie basée sur l'identité permettant de limiter l'accès à une ressource basée sur les balises de cette ressource, veuillez consulter Affichage des ressources en fonction des balises AWS IoT.

AWS IoT Rôles IAM

Un rôle IAM est une entité au sein de vous Compte AWS qui possède des autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec AWS IoT

Vous pouvez utiliser des informations d’identification temporaires pour vous connecter à l’aide de la fédération, endosser un rôle IAM ou encore pour endosser un rôle intercompte. Vous obtenez des informations d'identification de sécurité temporaires en appelant des opérations d' AWS STS API telles que AssumeRoleou GetFederationToken.

AWS IoT prend en charge l'utilisation d'informations d'identification temporaires.

Rôles liés à un service

Les rôles liés aux AWS services permettent aux services d'accéder aux ressources d'autres services pour effectuer une action en votre nom. Les rôles liés à un service s’affichent dans votre compte IAM et sont la propriété du service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

AWS IoT ne prend pas en charge les rôles liés à un service.

Rôles de service

Cette fonction permet à un service d’endosser une fonction du service en votre nom. Ce rôle autorise le service à accéder à des ressources d’autres services pour effectuer une action en votre nom. Les rôles de service s’affichent dans votre compte IAM et sont la propriété du compte. Cela signifie qu’un administrateur IAM peut modifier les autorisations associées à ce rôle. Toutefois, une telle action peut perturber le bon fonctionnement du service.

AWS IoT exemples de politiques basées sur l'identité

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou modifier les ressources AWS IoT . Ils ne peuvent pas non plus effectuer de tâches à l'aide de l' AWS API AWS Management Console AWS CLI, ou. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d’API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces stratégies aux utilisateurs ou aux groupes ayant besoin de ces autorisations.

Pour savoir comment créer une politique IAM basée sur l’identité à l’aide de ces exemples de documents de politique JSON, consultez Création de politiques dans l’onglet JSON dans le Guide de l’utilisateur IAM.

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console AWS IoT
- Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations
- Affichage des ressources en fonction des balises AWS IoT
- Afficher les ressources AWS IoT Device Advisor en fonction des balises

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer AWS IoT des ressources dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes

de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.

- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console AWS IoT

Pour accéder à la AWS IoT console, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails AWS IoT des ressources de votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Pour vous assurer que ces entités peuvent toujours utiliser la AWS IoT console, attachez également la politique AWS gérée suivante aux entités : `AWSIoTFullAccess` Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API que vous tentez d'effectuer.

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

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

Affichage des ressources en fonction des balises AWS IoT

Vous pouvez utiliser des conditions dans votre politique basée sur l'identité pour contrôler l'accès aux ressources AWS IoT en fonction des balises. Cet exemple montre comment créer une stratégie qui autorise l'affichage d'un objet. Toutefois, l'autorisation est accordée uniquement si la balise `Owner` de l'objet a pour valeur le nom d'utilisateur de cet utilisateur. Cette politique accorde également les autorisations nécessaires pour réaliser cette action sur la console.

JSON

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListBillingGroupsInConsole",
            "Effect": "Allow",
            "Action": "iot:ListBillingGroups",
            "Resource": "*"
        },
        {
            "Sid": "ViewBillingGroupsIfOwner",
            "Effect": "Allow",
           "Action": "iot:DescribeBillingGroup",
            "Resource": "arn:aws:iot:*:*:billinggroup/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Vous pouvez rattacher cette politique aux utilisateurs IAM de votre compte. Si un utilisateur nommé `richard-roe` tente de consulter un groupe AWS IoT de facturation, le groupe de facturation doit être étiqueté `Owner=richard-roe` ou`owner=richard-roe`. Dans le cas contraire, l'utilisateur se voit refuser l'accès. La clé de condition d'étiquette `Owner` correspond à la fois à `Owner` et à `owner`, car les noms de clé de condition ne sont pas sensibles à la casse. Pour plus d'informations, veuillez consulter la rubrique Éléments de stratégie JSON IAM : Condition dans le Guide de l'utilisateur IAM.

Afficher les ressources AWS IoT Device Advisor en fonction des balises

Vous pouvez utiliser des conditions dans votre politique basée sur l'identité pour contrôler l'accès aux ressources Device Advisor AWS IoT en fonction de balises. L'exemple suivant montre comment créer une politique permettant d'afficher une définition de suite particulière. Cependant, l'autorisation n'est accordée que si la balise de définition de suite a `SuiteType` définie sur la valeur `MQTT`. Cette politique accorde également les autorisations nécessaires pour réaliser cette action sur la console.

JSON

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

```
        {
            "Sid": "ViewSuiteDefinition",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:GetSuiteDefinition",
            "Resource": "arn:aws:iotdeviceadvisor:*:*:suitedefinition/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/SuiteType": "MQTT"}
            }
        }
    ]
}
```

AWS politiques gérées pour AWS IoT

Pour ajouter des autorisations aux utilisateurs, aux groupes et aux rôles, il est plus facile d'utiliser des politiques AWS gérées que de les rédiger vous-même. Il faut du temps et de l'expertise pour créer des politiques gérées par le client IAM qui ne fournissent à votre équipe que les autorisations dont elle a besoin. Pour démarrer rapidement, vous pouvez utiliser nos politiques AWS gérées. Ces politiques couvrent des cas d'utilisation courants et sont disponibles dans votre Compte AWS. Pour plus d'informations sur les politiques AWS gérées, voir les politiques AWS gérées dans le guide de l'utilisateur IAM.

AWS les services maintiennent et mettent à jour les politiques AWS gérées. Vous ne pouvez pas modifier les autorisations dans les politiques AWS gérées. Les services ajoutent occasionnellement des autorisations à une politique gérée par AWS pour prendre en charge de nouvelles fonctionnalités. Ce type de mise à jour affecte toutes les identités (utilisateurs, groupes et rôles) auxquelles la politique est attachée. Les services sont très susceptibles de mettre à jour une politique gérée par AWS quand une nouvelle fonctionnalité est lancée ou quand de nouvelles opérations sont disponibles. Les services ne suppriment pas les autorisations d'une politique AWS gérée. Les mises à jour des politiques n'endommageront donc pas vos autorisations existantes.

En outre, AWS prend en charge les politiques gérées pour les fonctions professionnelles qui couvrent plusieurs services. Par exemple, la politique ReadOnlyAccess AWS gérée fournit un accès en lecture seule à tous les AWS services et ressources. Quand un service lance une nouvelle fonctionnalité, AWS ajoute des autorisations en lecture seule pour les nouvelles opérations et ressources. Pour obtenir la liste des politiques de fonctions professionnelles et leurs descriptions, consultez la page politiques gérées par AWS pour les fonctions de tâche dans le Guide de l'utilisateur IAM.

> **Note**
>
> AWS IoT fonctionne avec les deux politiques AWS IoT et avec les politiques IAM. Cette rubrique traite uniquement des politiques IAM, qui définissent une action de politique pour les opérations d'API du plan de contrôle et du plan de données. Voir aussi AWS IoT Core politiques.

AWS politique gérée : AWSIo TConfig Accès

Vous pouvez associer la politique `AWSIoTConfigAccess` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'accéder à toutes les opérations de configuration AWS IoT . Cette stratégie peut affecter le traitement et le stockage des données. Pour consulter cette politique dans le AWS Management Console, voir AWSIoTConfigAccès.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `iot`— Récupérez AWS IoT des données et effectuez des actions de configuration IoT.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AcceptCertificateTransfer",
                "iot:AddThingToThingGroup",
```

```
                "iot:AssociateTargetsWithJob",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CancelCertificateTransfer",
                "iot:CancelJob",
                "iot:CancelJobExecution",
                "iot:ClearDefaultAuthorizer",
                "iot:CreateAuthorizer",
                "iot:CreateCertificateFromCsr",
                "iot:CreateJob",
                "iot:CreateKeysAndCertificate",
                "iot:CreateOTAUpdate",
                "iot:CreatePolicy",
                "iot:CreatePolicyVersion",
                "iot:CreateRoleAlias",
                "iot:CreateStream",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:CreateThingType",
                "iot:CreateTopicRule",
                "iot:DeleteAuthorizer",
                "iot:DeleteCACertificate",
                "iot:DeleteCertificate",
                "iot:DeleteJob",
                "iot:DeleteJobExecution",
                "iot:DeleteOTAUpdate",
                "iot:DeletePolicy",
                "iot:DeletePolicyVersion",
                "iot:DeleteRegistrationCode",
                "iot:DeleteRoleAlias",
                "iot:DeleteStream",
                "iot:DeleteThing",
                "iot:DeleteThingGroup",
                "iot:DeleteThingType",
                "iot:DeleteTopicRule",
                "iot:DeleteV2LoggingLevel",
                "iot:DeprecateThingType",
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
```

```
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachPrincipalPolicy",
                "iot:DetachThingPrincipal",
                "iot:DisableTopicRule",
                "iot:EnableTopicRule",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
```

```
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:RegisterCACertificate",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RejectCertificateTransfer",
                "iot:RemoveThingFromThingGroup",
                "iot:ReplaceTopicRule",
                "iot:SearchIndex",
                "iot:SetDefaultAuthorizer",
                "iot:SetDefaultPolicyVersion",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:SetV2LoggingOptions",
                "iot:StartThingRegistrationTask",
                "iot:StopThingRegistrationTask",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:TransferCertificate",
                "iot:UpdateAuthorizer",
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:UpdateEventConfigurations",
                "iot:UpdateIndexingConfiguration",
                "iot:UpdateRoleAlias",
                "iot:UpdateStream",
                "iot:UpdateThing",
                "iot:UpdateThingGroup",
                "iot:UpdateThingGroupsForThing",
                "iot:UpdateAccountAuditConfiguration",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DeleteAccountAuditConfiguration",
                "iot:StartOnDemandAuditTask",
                "iot:CancelAuditTask",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:CreateScheduledAudit",
```

```
                "iot:UpdateScheduledAudit",
                "iot:DeleteScheduledAudit",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:CreateSecurityProfile",
                "iot:DescribeSecurityProfile",
                "iot:UpdateSecurityProfile",
                "iot:DeleteSecurityProfile",
                "iot:AttachSecurityProfile",
                "iot:DetachSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politique gérée : AWSIo TConfig ReadOnlyAccess

Vous pouvez associer la politique `AWSIoTConfigReadOnlyAccess` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'accéder en lecture seule à toutes les opérations de configuration AWS IoT . Pour consulter cette politique dans le AWS Management Console, voir AWSIoTConfigReadOnlyAccess.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `iot` – Effectuez des opérations en lecture seule sur les actions de configuration IoT.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
```

```
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:SearchIndex",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:DescribeSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politique gérée : AWSIo TData Accès

Vous pouvez associer la politique `AWSIoTDataAccess` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'accéder à toutes les opérations de AWS IoT données. Les opérations sur les données envoient des données via les protocoles MQTT ou HTTP. Pour afficher cette politique dans AWS Management Console, veuillez consulter `AWSIoTDataAccess`.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `iot`— Récupérez AWS IoT les données et autorisez un accès complet aux actions AWS IoT de messagerie.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow",
                "iot:ListNamedShadowsForThing"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS politique gérée : AWSIo TFull Accès

Vous pouvez associer la politique `AWSIoTFullAccess` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'accéder à toutes les opérations de configuration et de messagerie AWS IoT . Pour consulter cette politique dans le AWS Management Console, voir `AWSIoTFullAccess`.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `iot`— Récupérez AWS IoT les données et autorisez un accès complet aux actions AWS IoT de configuration et de messagerie.
- `iotjobsdata`— Récupérez les données des AWS IoT tâches et autorisez un accès complet aux opérations de l'API du plan de données des AWS IoT tâches.

JSON

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

AWS politique gérée : AWSIo TLogging

Vous pouvez associer la politique `AWSIoTLogging` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'accéder à la création de groupes Amazon CloudWatch Logs et à la diffusion des journaux vers les groupes. Cette politique est liée à votre rôle de CloudWatch journalisation. Pour consulter cette politique dans le AWS Management Console, voir `AWSIoTLogging`.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `logs`— Récupère CloudWatch les journaux. Permet également de créer des groupes de CloudWatch journaux et de diffuser des journaux vers les groupes.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DeleteLogStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS politique gérée : AWSIo TOTAUpdate

Vous pouvez associer la politique `AWSIoTOTAUpdate` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'accéder à la création de AWS IoT tâches, aux tâches de signature de AWS IoT code et à la description des tâches de signature de AWS code. Pour consulter cette politique dans le AWS Management Console, voir `AWSIoTOTAUpdate.`

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `iot`— Créez des AWS IoT tâches et des tâches de signature de code.
- `signer`— Réalise la création de tâches de signature de AWS code.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateJob",
            "signer:DescribeSigningJob"
        ],
        "Resource": "*"
    }
}
```

AWS politique gérée : AWSIo TRule Actions

Vous pouvez associer la politique `AWSIoTRuleActions` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'accéder à toutes les informations Service AWS prises en charge dans les actions de AWS IoT règles. Pour consulter cette politique dans le AWS Management Console, voir `AWSIoTRuleActions`.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `iot` - Effectuez des actions pour publier des messages d’action de la règle.
- `dynamodb` - Insérez un message dans un tableau DynamoDB ou divisez un message en plusieurs colonnes dans un tableau DynamoDB.
- `s3` - Stockez un objet dans un compartiment Amazon S3.
- `kinesis` - Envoyez un message à un objet de flux Amazon Kinesis.
- `firehose`- Insérez un enregistrement dans un objet de flux Firehose.
- `cloudwatch`- Modifiez l'état de l' CloudWatch alarme ou envoyez les données du message au CloudWatch système métrique.
- `sns` - Effectuez une opération pour publier une notification à l'aide d'Amazon SNS. Cette opération est limitée aux rubriques AWS IoT SNS.
- `sqs` - Insérez un message à ajouter à la file d'attente SQS.
- `es`- Envoyez un message au OpenSearch service Service.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "dynamodb:PutItem",
            "kinesis:PutRecord",
            "iot:Publish",
            "s3:PutObject",
            "sns:Publish",
            "sqs:SendMessage*",
            "cloudwatch:SetAlarmState",
```

```
            "cloudwatch:PutMetricData",
            "es:ESHttpPut",
            "firehose:PutRecord"
        ],
        "Resource": "*"
    }
}
```

AWS politique gérée : AWSIo TThings enregistrement

Vous pouvez associer la politique `AWSIoTThingsRegistration` à vos identités IAM.

Cette politique accorde les autorisations d'identité associées qui permettent d'enregistrer des éléments en masse à l'aide de l'API `StartThingRegistrationTask`. Cette stratégie peut affecter le traitement et le stockage des données. Pour consulter cette politique dans le AWS Management Console, voir `AWSIoTThingsRegistration`.

Détails de l’autorisation

Cette politique inclut les autorisations suivantes.

- `iot` - Effectuez des actions pour créer des éléments et joindre des politiques et des certificats lors d'un enregistrement groupé.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
```

```
                "iot:CreateCertificateFromCsr",
                "iot:CreatePolicy",
                "iot:CreateThing",
                "iot:DescribeCertificate",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachThingPrincipal",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RemoveThingFromThingGroup",
                "iot:UpdateCertificate",
                "iot:UpdateThing",
                "iot:UpdateThingGroupsForThing",
                "iot:AddThingToBillingGroup",
                "iot:DescribeBillingGroup",
                "iot:RemoveThingFromBillingGroup"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS IoT mises à jour des politiques AWS gérées

Consultez les détails des mises à jour des politiques AWS gérées AWS IoT depuis que ce service a commencé à suivre ces modifications. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS sur la page Historique du AWS IoT document.

Modification	Description	Date
AWSIoTFullAccès — Mise à jour d'une politique existante	AWS IoT a ajouté de nouvelles autorisations pour permettre aux utilisateurs d'accéder aux opérations de l'API du plan de données AWS IoT Jobs à l'aide du protocole HTTP. Un nouveau préfixe de politique IAM vous fournit un contrôle d'accès plus précis pour accéder aux points de terminaison du plan de données AWS IoT Jobs. `iotjobsdata:` Pour les opérations de l'API du plan de contrôle, vous utilisez toujours le préfixe `iot:`. Pour de plus amples informations, veuillez consulter AWS IoT Core politiques pour le protocole HTTPS.	11 mai 2022
AWS IoT a commencé à suivre les modifications	AWS IoT a commencé à suivre les modifications apportées AWS à ses politique s gérées.	11 mai 2022

Résolution des problèmes AWS IoT d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS IoT IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans AWS IoT
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS IoT ressources

Je ne suis pas autorisé à effectuer une action dans AWS IoT

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit lorsque l'utilisateur IAM, `mateojackson` , tente d'utiliser la console pour afficher les détails d'une ressource d'objet, mais ne dispose pas des autorisations `iot:DescribeThing` nécessaires.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iot:DescribeThing on resource: MyIoTThing
```

Dans ce cas, la politique qui s'applique à l'utilisateur `mateojackson` doit être mise à jour pour autoriser l'accès à la ressource de l'objet à l'aide de l'action `iot:DescribeThing`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Utilisation de AWS IoT Device Advisor

Si vous utilisez AWS IoT Device Advisor, l'exemple d'erreur suivant se produit lorsque l'utilisateur `mateojackson` essaie d'utiliser la console pour consulter les détails relatifs à la définition d'une suite, mais qu'il n'en a pas les `iotdeviceadvisor:GetSuiteDefinition` autorisations.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotdeviceadvisor:GetSuiteDefinition on resource: MySuiteDefinition
```

Dans ce cas, la politique de l'utilisateur `mateojackson` doit être mise à jour pour autoriser l'accès à la ressource *MySuiteDefinition* à l'aide de l'action `iotdeviceadvisor:GetSuiteDefinition`.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter `iam:PassRole` l'action, vos stratégies doivent être mises à jour afin de vous permettre de transmettre un rôle à AWS IoT.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans AWS IoT. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes AWS IoT ressources

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si ces fonctionnalités sont prises AWS IoT en charge, consultezComment AWS IoT fonctionne avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.

- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Journalisation et surveillance

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS IoT et des performances de vos AWS solutions. Vous devez collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de pouvoir corriger plus facilement une défaillance multipoint, le cas échéant. Pour de plus amples informations sur les procédures de journalisation et de surveillance, veuillez consulter Surveillance AWS IoT

Outils de supervision

AWS fournit des outils que vous pouvez utiliser pour surveiller AWS IoT. Vous pouvez configurer certains de ces outils afin qu'ils effectuent la surveillance à votre place. Une intervention manuelle est nécessaire pour certains outils. Nous vous recommandons d'automatiser le plus possible les tâches de supervision.

Outils de surveillance automatique

Vous pouvez utiliser les outils de surveillance automatique suivants pour surveiller AWS IoT et signaler tout problème :

- Amazon CloudWatch Alarms — Surveillez une seule métrique sur une période que vous spécifiez et effectuez une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. L'action est une notification envoyée à une rubrique Amazon Simple Notification Service (Amazon SNS) ou à une politique Amazon EC2 Auto Scaling. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier. L'état doit avoir changé et avoir été maintenu pendant un nombre de périodes spécifié.

Pour de plus amples informations, veuillez consulter Surveillez les AWS IoT alarmes et les métriques à l'aide d'Amazon CloudWatch.

- Amazon CloudWatch Logs — Surveillez, stockez et accédez à vos fichiers journaux depuis AWS CloudTrail ou d'autres sources. Amazon CloudWatch Logs vous permet également de voir les étapes critiques des scénarios de test AWS IoT Device Advisor, les événements générés et les messages MQTT envoyés depuis vos appareils ou AWS IoT Core pendant l'exécution des tests. Ces journaux permettent de déboguer et de prendre des actions correctives sur vos appareils. Pour plus d'informations, consultez Surveiller AWS IoT à l'aide CloudWatch des journaux Pour plus d'informations sur l'utilisation d'Amazon CloudWatch, consultez la section Surveillance des fichiers journaux dans le guide de CloudWatch l'utilisateur Amazon.
- Amazon CloudWatch Events : associez les événements et acheminez-les vers une ou plusieurs fonctions ou flux cibles afin d'apporter des modifications, de recueillir des informations d'état et de prendre des mesures correctives. Pour plus d'informations, consultez la section Qu'est-ce qu'Amazon CloudWatch Events dans le guide de CloudWatch l'utilisateur Amazon.
- AWS CloudTrail Surveillance des journaux : partagez les fichiers journaux entre les comptes, surveillez les fichiers CloudTrail CloudWatch journaux en temps réel en les envoyant à Logs, écrivez des applications de traitement des journaux en Java et vérifiez que vos fichiers journaux n'ont pas changé après leur livraison par CloudTrail. Pour plus d'informations, consultez également Enregistrement des appels AWS IoT d'API à l'aide de AWS CloudTrail la section Utilisation des fichiers CloudTrail journaux dans le guide de AWS CloudTrail l'utilisateur.

Outils de surveillance manuelle

Une autre partie importante de la surveillance AWS IoT consiste à surveiller manuellement les éléments non couverts par les CloudWatch alarmes. Le tableau de bord AWS IoT CloudWatch, et les autres tableaux de bord de la console de AWS service, fournissent une at-a-glance vue d'ensemble de l'état de votre AWS environnement. Nous vous recommandons de vérifier également les fichiers journaux AWS IoT.

- AWS IoT le tableau de bord affiche :
  - Certificats CA
  - Certificats
  - Stratégies
  - Règles
  - Objets

- CloudWatch la page d'accueil montre :
  - Alarmes et statuts en cours.
  - Graphiques des alarmes et des ressources.
  - Statut d'intégrité du service.

  Vous pouvez utiliser CloudWatch pour effectuer les opérations suivantes :
  - Créer des tableaux de bord personnalisés pour surveiller les services de votre choix.
  - Données de métriques de graphiques pour résoudre les problèmes et découvrir les tendances.
  - Recherchez et parcourez tous les indicateurs de vos AWS ressources.
  - Créer et Modifier des alarmes pour être informé des problèmes.

Validation de conformité pour AWS IoT Core

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Conformité et gouvernance de la sécurité : ces guides de mise en œuvre de solutions traitent des considérations architecturales et fournissent les étapes à suivre afin de déployer des fonctionnalités de sécurité et de conformité.
- Référence des services éligibles HIPAA : liste les services éligibles HIPAA. Tous ne Services AWS sont pas éligibles à la loi HIPAA.
- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- AWS Guides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques en matière

de sécurisation Services AWS et décrivent les directives relatives aux contrôles de sécurité dans de nombreux cadres (notamment le National Institute of Standards and Technology (NIST), le Payment Card Industry Security Standards Council (PCI) et l'Organisation internationale de normalisation (ISO)).

- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Cela Service AWS fournit une vue complète de votre état de sécurité interne AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty — Cela Service AWS détecte les menaces potentielles qui pèsent sur vos charges de travail Comptes AWS, vos conteneurs et vos données en surveillant votre environnement pour détecter toute activité suspecte et malveillante. GuardDuty peut vous aider à répondre à diverses exigences de conformité, telles que la norme PCI DSS, en répondant aux exigences de détection des intrusions imposées par certains cadres de conformité.
- AWS Audit Manager— Cela vous Service AWS permet d'auditer en permanence votre AWS utilisation afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

La résilience au cœur de AWS l'IoT

L'infrastructure AWS mondiale est construite autour de Région AWS s et de zones de disponibilité. Région AWS s fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les Région AWS zones de disponibilité et les zones de disponibilité, consultez la section Infrastructure AWS globale.

AWS IoT Core enregistre les informations relatives à vos appareils dans le registre des appareils. Il stocke également les certificats d'autorité de certification, les certificats d'appareil et les données

de shadow d'appareil. En cas de panne matérielle ou réseau, ces données sont automatiquement répliquées dans les zones de disponibilité mais pas dans les régions.

AWS IoT Core publie des événements MQTT lorsque le registre des appareils est mis à jour. Vous pouvez utiliser ces messages pour sauvegarder vos données de registre et les enregistrer quelque part, par exemple dans un tableau DynamoDB. Vous êtes responsable de la sauvegarde des certificats AWS IoT Core créés pour vous ou ceux que vous créez vous-même. Device Shadow stocke les données d'état relatives à vos appareils et peut être renvoyée lorsqu'un appareil revient en ligne. AWS IoT Device Advisor stocke les informations relatives à la configuration de votre suite de tests. Ces données sont automatiquement répliquées en cas de défaillance du matériel ou du réseau.

AWS IoT Core les ressources sont spécifiques à une région et ne sont pas répliquées à Régions AWS moins que vous ne le fassiez spécifiquement.

Pour plus d'informations sur les bonnes pratiques de Sécurité, consultez Bonnes pratiques en matière de sécurité dans AWS IoT Core.

Utilisation AWS IoT Core avec des points de terminaison VPC d'interface

Avec AWS IoT Core, vous pouvez créer des points de terminaison de données IoT au sein de votre cloud privé virtuel (VPC) en utilisant des points de terminaison VPC d'interface. Les points de terminaison VPC d'interface sont alimentés par AWS PrivateLink une AWS technologie que vous pouvez utiliser pour accéder aux services exécutés à l'aide AWS d'adresses IP privées. Pour en savoir plus, consultez Amazon Virtual Private Cloud.

Pour connecter des appareils sur le terrain sur des réseaux distants, tels qu'un réseau d'entreprise, à votre Amazon VPC, reportez-vous aux options répertoriées dans la matrice de connectivité Network-to-Amazon VPC.

Table des matières

- Création de points de terminaison VPC pour le plan de données AWS IoT Core
- Création de points de terminaison d'un VPC pour le fournisseur d'informations d'identification AWS IoT Core
- Création d'un point de terminaison d'interface Amazon VPC
- Configuration d'une zone hébergée privée
- Contrôle de l'accès aux points de AWS IoT Core terminaison via VPC

- Limites
- Dimensionnement des points de terminaison VPC avec AWS IoT Core
- Utiliser des domaines personnalisés avec des points de terminaison d'un VPC
- Disponibilité des points de terminaison VPC pour AWS IoT Core

Création de points de terminaison VPC pour le plan de données AWS IoT Core

Vous pouvez créer un point de terminaison VPC pour l'API du plan de AWS IoT Core données afin de connecter vos appareils à des AWS IoT services et à d'autres AWS services. Pour commencer à utiliser les points de terminaison VPC, créez un point de terminaison VPC d'interface et sélectionnez-le comme service. AWS IoT Core AWS Si vous utilisez la CLI, appelez d'abord describe-vpc-endpoint-servicespour vous assurer que vous choisissez une zone de disponibilité où elle AWS IoT Core est présente dans votre environnement Région AWS. Par exemple, dans us-east-1, cette commande ressemblerait à :

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.data
```

> **Note**
>
> La fonctionnalité VPC permettant de créer automatiquement un enregistrement DNS est désactivée. Pour vous connecter à ces points de terminaison, vous devez créer manuellement un enregistrement DNS privé. Pour plus d'informations sur les enregistrements DNS d'un VPC privé, consultez DNS privé pour les points de terminaison d'interface. Pour plus d'informations sur les limites du AWS IoT Core VPC, consultez. Limites

Pour connecter les clients MQTT aux interfaces du point de terminaison du VPC :

- Vous devez créer manuellement des enregistrements DNS dans une zone hébergée privée attachée à votre VPC. Pour commencer, consultez Création d'une zone hébergée privée.

- Dans votre zone hébergée privée, créez un enregistrement d'alias pour chaque adresse IP d'interface réseau élastique pour le point de terminaison VPC. Si vous disposez de plusieurs interfaces réseau IPs pour plusieurs points de terminaison VPC, créez des enregistrements DNS

pondérés avec des pondérations égales pour tous les enregistrements pondérés. Ces adresses IP sont disponibles à partir de l'appel d'DescribeNetworkInterfacesAPI lorsqu'elles sont filtrées par l'ID du point de terminaison VPC dans le champ de description.

Consultez les instructions détaillées ci-dessous pour créer un point de terminaison d'interface Amazon VPC et configurer une zone hébergée privée pour le plan de AWS IoT Core données.

Création de points de terminaison d'un VPC pour le fournisseur d'informations d'identification AWS IoT Core

Vous pouvez créer un point de terminaison VPC pour que le fournisseur AWS IoT Core d'informations d'identification connecte les appareils à l'aide de l'authentification basée sur un certificat client et obtienne des informations d' AWS identification temporaires au AWS format Signature Version 4. Pour commencer à utiliser les points de terminaison VPC pour le fournisseur AWS IoT Core d'informations d'identification, exécutez la commande create-vpc-endpointCLI pour créer un point de terminaison VPC d'interface et sélectionnez AWS IoT Core le fournisseur d'informations d'identification comme service. AWS Pour vous assurer que vous choisissez une zone de disponibilité où elle AWS IoT Core est présente dans votre environnement Région AWS, vous devez d'abord exécuter la describe-vpc-endpoint-servicescommande. Par exemple, dans us-east-1, cette commande ressemblerait à :

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.iot.credentials
```

> **Note**
>
> La fonctionnalité VPC permettant de créer automatiquement un enregistrement DNS est désactivée. Pour vous connecter à ces points de terminaison, vous devez créer manuellement un enregistrement DNS privé. Pour plus d'informations sur les enregistrements DNS d'un VPC privé, consultez DNS privé pour les points de terminaison d'interface. Pour plus d'informations sur les limites du AWS IoT Core VPC, consultez. Limites

Pour connecter les clients HTTP aux interfaces du point de terminaison du VPC :

- Vous devez créer manuellement des enregistrements DNS dans une zone hébergée privée attachée à votre VPC. Pour commencer, consultez Création d'une zone hébergée privée.

- Dans votre zone hébergée privée, créez un enregistrement d'alias pour chaque adresse IP d'interface réseau élastique pour le point de terminaison VPC. Si vous disposez de plusieurs interfaces réseau IPs pour plusieurs points de terminaison VPC, créez des enregistrements DNS pondérés avec des pondérations égales pour tous les enregistrements pondérés. Ces adresses IP sont disponibles à partir de l'appel d'DescribeNetworkInterfacesAPI lorsqu'elles sont filtrées par l'ID du point de terminaison VPC dans le champ de description.

Consultez les instructions détaillées ci-dessous pour créer un point de terminaison d'interface Amazon VPC et configurer une zone hébergée privée pour le fournisseur AWS IoT Core d'informations d'identification.

Création d'un point de terminaison d'interface Amazon VPC

Vous pouvez créer un point de terminaison VPC d'interface pour vous connecter aux AWS services alimentés par. AWS PrivateLink Utilisez la procédure suivante pour créer un point de terminaison VPC d'interface qui se connecte au plan de AWS IoT Core données ou au fournisseur AWS IoT Core d'informations d'identification. Pour plus d'informations, consultez Accéder à un AWS service à l'aide d'un point de terminaison VPC d'interface.

> **Note**
>
> Les processus de création d'un point de terminaison d'interface Amazon VPC pour le plan de AWS IoT Core données et le fournisseur AWS IoT Core d'informations d'identification sont similaires, mais vous devez apporter des modifications spécifiques au point de terminaison pour que la connexion fonctionne.

Pour créer un point de terminaison d'un VPC d'interface à l'aide du point de terminaison du VPC de la console

1. Accédez au point de terminaison d'un VPC de la console , sous Virtual private cloud dans le menu de gauche, choisissez point de terminaison puis Créer un point de terminaison .
2. Dans la page Create Endpoint (Créer un point de terminaison, spécifiez les informations suivantes.
   - Choisissez Service AWS s pour la catégorie de service .
   - Pour Nom du service, effectuez une recherche en saisissant le mot-clé `iot`. Dans la liste des services `iot` affichés, choisissez le point de terminaison.

Si vous créez un point de terminaison VPC pour le plan de AWS IoT Core données, choisissez le point de terminaison de l'API du plan de AWS IoT Core données pour votre région. Le format du nom du point de terminaison est `com.amazonaws.region.iot.data`.

Si vous créez un point de terminaison VPC pour le fournisseur AWS IoT Core d'informations d'identification, choisissez le point de terminaison du fournisseur AWS IoT Core d'informations d'identification pour votre région. Le format du nom du point de terminaison est `com.amazonaws.region.iot.credentials`.

> **Note**
>
> Le nom du service pour le plan de AWS IoT Core données dans la région de Chine sera au format suivant`cn.com.amazonaws.region.iot.data`. La création de points de terminaison VPC pour le fournisseur AWS IoT Core d'informations d'identification n'est pas prise en charge dans la région de la Chine.

- Pour le VPC et les sous-réseaux, choisissez le VPC dans lequel vous souhaitez créer le point de terminaison et les zones de disponibilité (AZs) dans lesquelles vous souhaitez créer le réseau de points de terminaison.
- Pour Activer le nom DNS, assurez-vous que Activer pour ce point de terminaison n'est pas sélectionné. Ni le plan AWS IoT Core de données ni le fournisseur AWS IoT Core d'informations d'identification ne prennent encore en charge les noms DNS privés.
- Pour (Groupe de sécurité), sélectionnez les groupes de sécurité que vous souhaitez associer aux interfaces réseau des points de terminaison.
- En option, vous pouvez ajouter ou supprimer des balises. Les balises sont des paires nom-valeur que vous utilisez pour associer à votre point de terminaison.

3. Pour créer votre point de terminaison VPC, choisissez Créer un point de terminaison.

Après avoir créé le AWS PrivateLink point de terminaison, dans l'onglet Détails de votre point de terminaison, vous verrez une liste de noms DNS. Vous pouvez utiliser l'un de ces noms DNS que vous avez créés dans cette section pour configurer votre zone hébergée privée.

Configuration d'une zone hébergée privée

Vous pouvez utiliser l'un de ces noms DNS que vous avez créés dans la section précédente pour configurer votre zone hébergée privée.

Pour le plan AWS IoT Core de données

Le nom DNS doit être le nom de configuration de votre domaine ou votre point de terminaison `IoT:Data-ATS`. Un exemple de nom DNS peut être : *`xxx`*`-ats.data.iot.`*`region`*`.amazonaws.com`.

Pour le fournisseur AWS IoT Core d'informations d'identification

Le nom DNS doit être votre point de terminaison `iot:CredentialProvider`. Un exemple de nom DNS peut être : *`xxxx`*`.credentials.iot.`*`region`*`.amazonaws.com`.

> **Note**
>
> Les processus de configuration de la zone hébergée privée pour le plan de AWS IoT Core données et le fournisseur AWS IoT Core d'informations d'identification sont similaires, mais vous devez apporter des modifications spécifiques au point de terminaison pour que la connexion fonctionne.

Créer une zone hébergée privée

Pour créer une zone hébergée privée à l'aide de la console Route 53

1. Accédez à la console Zones hébergées Route 53 et choisissez Créer une zone hébergée.
2. Dans la page Créer une zone hébergée, spécifiez les informations suivantes.
   - Pour Nom de domaine, entrez l'adresse du point de terminaison de votre `iot:Data-ATS` ou de votre point de terminaison `iot:CredentialProvider`. La commande CLI AWS suivante montre comment obtenir le point de terminaison via un réseau public : `aws iot describe-endpoint --endpoint-type iot:Data-ATS`, ou `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`.

     > **Note**
     >
     > Si vous utilisez des domaines personnalisés, consultez Utilisation de domaines personnalisés avec des points de terminaison d'un VPC. Les domaines personnalisés ne sont pas pris en charge par le fournisseur d' AWS IoT Core informations d'identification.

   - Pour Type, choisissez Zone hébergée privée.

- En option, vous pouvez ajouter ou supprimer des balises à associer à votre zone hébergée.

3. Pour créer votre zone hébergée privée, choisissez Créer une zone hébergée.

Pour plus d'informations, consultez Création d'une zone hébergée privée.

Créer un enregistrement

Après avoir créé une zone hébergée privée, vous pouvez créer un enregistrement indiquant au DNS sur la façon dont vous souhaitez acheminer le trafic vers ce domaine.

Pour créer un enregistrement

1. Dans la liste des zones hébergées affichée, choisissez la zone hébergée privée que vous avez créée précédemment et sur Créer un enregistrement.
2. Utilisez la méthode d'assistance pour créer l'enregistrement. Si la console vous présente la méthode de Création rapide, choisissez Passer à l'assistant.
3. Choisissez Routage simple pour Stratégie de routage, puis sur Suivant.
4. Dans la page Configurer les enregistrements, choisissez Définir un enregistrement simple.
5. Dans la page Définir un enregistrement simple :
   - Pour Nom de l'enregistrement, saisissez le point de terminaison `iot:Data-ATS` ou le point de terminaison `iot:CredentialProvider`. Cela doit être identique au nom de la zone hébergée privée.
   - Pour Type d'enregistrement, conservez la valeur à `A - Routes traffic to an IPv4 address and some AWS resources`.
   - Pour Valeur/Route du trafic vers , choisissez Alias vers le point de terminaison d'un VPC. Choisissez ensuite votre Région, puis Point de terminaison que vous avez créé précédemment, comme décrit dans la liste des points de terminaison ??? affichés.
6. Choisissez Définir un enregistrement simple pour créer votre enregistrement.

Contrôle de l'accès aux points de AWS IoT Core terminaison via VPC

Vous pouvez restreindre l'accès aux appareils AWS IoT Core pour qu'il soit autorisé uniquement via le point de terminaison VPC en utilisant les clés contextuelles de condition VPC. AWS IoT Core prend en charge les clés de contexte liées au VPC suivantes :

- SourceVpc

- SourceVpce
- VPCSourceIP

> **Note**
>
> AWS IoT Core ne prend pas en charge les politiques de points de terminaison pour les points de terminaison VPC.

Par exemple, la politique suivante accorde l'autorisation de se connecter à AWS IoT Core l'aide d'un ID client correspondant au nom de l'objet et de publier sur n'importe quelle rubrique préfixée par le nom de l'objet, à condition que l'appareil se connecte à un point de terminaison VPC avec un identifiant de point de terminaison VPC particulier. Cette stratégie refuserait les tentatives de connexion à votre point de terminaison de données IoT public.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }

        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
```

```
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Limites

Les points de terminaison d'un VPC sont actuellement pris en charge uniquement pour les AWS IoT Core points de terminaison de données et AWS IoT Core les points de terminaison du fournisseur d'informations d'identification. Les points de terminaison VPC ne sont pas pris en charge pour les points de terminaison FIPS (Federal Information Processing Standard).

Limites des points de terminaison d'un VPC de données IoT

Cette section couvre les limitations des points de terminaison d'un VPC de données IoT.

- Les périodes de maintien en vie MQTT sont limitées à 230 secondes. Les périodes de maintien en vie plus longues seront automatiquement réduites à 230 secondes.
- Chaque point de terminaison d'un VPC prend en charge un total de 100 000 appareils connectés simultanément. Si vous avez besoin de plus de connexions, consultez Dimensionnement des points de terminaison VPC avec AWS IoT Core.
- Les points de terminaison VPC ne prennent en charge IPv4 que le trafic.
- Les points de terminaison d'un VPC serviront uniquement les certificats ATS, à l'exception des domaines personnalisés.
- Les politiques de point de terminaison d'un VPC ne sont pas prises en charge.
- Pour les points de terminaison VPC créés pour le plan de AWS IoT Core données, l'utilisation d'enregistrements DNS publics zonaux ou régionaux AWS IoT Core n'est pas prise en charge.

Limites des points de terminaison du fournisseur d'informations d'identification

Cette section couvre les limitations des points de terminaison d'un VPC du fournisseur d'informations d'identification.

- Les points de terminaison VPC ne prennent en charge IPv4 que le trafic.

- Les points de terminaison d'un VPC serviront uniquement les certificats ATS.
- Les politiques de point de terminaison d'un VPC ne sont pas prises en charge.
- Les domaines personnalisés ne sont pas pris en charge pour les points de terminaison du fournisseur d'informations d'identification.
- Pour les points de terminaison VPC créés pour le fournisseur AWS IoT Core d'informations d'identification, l'utilisation d'enregistrements DNS publics zonaux ou régionaux AWS IoT Core n'est pas prise en charge.

Dimensionnement des points de terminaison VPC avec AWS IoT Core

AWS IoT Core Les points de terminaison VPC d'interface sont limités à 100 000 appareils connectés sur un seul point de terminaison d'interface. Si votre cas d'utilisation nécessite davantage de connexions simultanées au courtier, nous vous recommandons d'utiliser plusieurs points de terminaison d'un VPC et de router manuellement vos appareils via vos points de terminaison d'interface. Lorsque vous créez des enregistrements DNS privés pour acheminer le trafic vers vos points de terminaison d'un VPC, assurez-vous de créer autant d'enregistrements pondérés que vous disposez de points de terminaison de VPC pour répartir le trafic sur vos multiples points de terminaison.

Utiliser des domaines personnalisés avec des points de terminaison d'un VPC

Si vous souhaitez utiliser des domaines personnalisés avec des points de terminaison d'un VPC, vous devez créer vos enregistrements de nom de domaine personnalisés dans une zone hébergée privée et créer des enregistrements de routage dans Route53. Pour plus d'informations, consultez Création d'une zone hébergée privée.

> **Note**
>
> Les domaines personnalisés ne sont pris en charge que pour les points de terminaison de AWS IoT Core données.

Disponibilité des points de terminaison VPC pour AWS IoT Core

AWS IoT Core Les points de terminaison VPC d'interface sont disponibles dans toutes AWS IoT Core les régions prises en charge. AWS IoT Core Les points de terminaison VPC d'interface pour le

fournisseur AWS IoT Core d'informations d'identification ne sont pas pris en charge dans la région Chine et. AWS GovCloud (US) Regions

Sécurité de l'infrastructure dans AWS IoT

En tant qu'ensemble de services gérés, AWS IoT il est protégé par les procédures de sécurité du réseau AWS mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder AWS IoT via le réseau. Les clients doivent prendre en charge le protocole TLS (Transport Layer Security) 1.2 ou version ultérieure. Les clients doivent aussi prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La plupart des systèmes modernes telles que Java 7 et versions ultérieures prennent en charge ces modes. Pour de plus amples informations, veuillez consulter Sûreté des transports dans AWS IoT Core.

Les demandes doivent être signées à l'aide d'un identifiant de clé d'accès et d'une clé d'accès secrète associée à un mandataire IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Surveillance de la sécurité des flottes de production ou des appareils avec Core AWS IoT

Les flottes IoT sont composées d'un grand nombre d'appareils disposant de capacités diverses, d'une durée de vie longue et qui sont répartis géographiquement. Ces caractéristiques rendent la configuration du parc complexe et sujette aux erreurs. Les appareils étant souvent limités en puissance de calcul, de mémoire et de capacités de stockage, l'utilisation du chiffrement et d'autres formes de sécurité sur les appareils eux-mêmes s'en trouve limitée. En outre, les appareils utilisent souvent des logiciels aux vulnérabilités connues. Ces facteurs font des parcs IoT une cible attractive pour les pirates informatiques et rend difficile la sécurisation de votre parc sur une base permanente.

AWS IoT Device Defender répond à ces défis en fournissant des outils permettant d'identifier les problèmes de sécurité et les écarts par rapport aux meilleures pratiques. Vous pouvez utiliser AWS IoT Device Defender pour analyser, auditer et surveiller les appareils connectés afin de détecter les comportements anormaux et d'atténuer les risques de sécurité. AWS IoT Device Defender peut auditer les flottes d'appareils pour s'assurer qu'elles respectent les meilleures pratiques de sécurité

et détecter les comportements anormaux sur les appareils. Cela permet d'appliquer des politiques de sécurité cohérentes à l'ensemble de votre parc d' AWS IoT appareils et de réagir rapidement lorsque des appareils sont compromis. Pour de plus amples informations, veuillez consulter AWS IoT Device Defender.

AWS IoT Device Advisor diffuse les mises à jour et corrige votre parc en fonction de vos besoins. AWS IoT Device Advisor met automatiquement à jour les scénarios de test. Les cas de test que vous sélectionnez sont toujours avec la dernière version. Pour de plus amples informations, veuillez consulter Device Advisor.

Bonnes pratiques en matière de sécurité dans AWS IoT Core

Cette section contient des informations sur les meilleures pratiques de sécurité pour AWS IoT Core. Pour plus d'informations sur les règles de sécurité pour les solutions Industrial IoT, consultez Dix règles d'or de sécurité pour les solutions Industrial IoT.

Protection des connexions MQTT dans AWS IoT

AWS IoT Coreest un service cloud géré qui permet aux appareils connectés d'interagir avec des applications cloud et d'autres appareils facilement et en toute sécurité. AWS IoT Core supporte HTTP et MQTT WebSocket, un protocole de communication léger spécialement conçu pour tolérer les connexions intermittentes. Si vous vous connectez AWS IoT via MQTT, chacune de vos connexions doit être associée à un identifiant appelé ID client. Le client MQTT identifie de IDs manière unique les connexions MQTT. Si une nouvelle connexion est établie à l'aide d'un ID client déjà réclamé pour une autre connexion, le courtier de AWS IoT messages abandonne l'ancienne connexion pour autoriser la nouvelle connexion. Le client IDs doit être unique au sein de chacun Compte AWS d'entre eux Région AWS. Cela signifie que vous n'avez pas besoin d'appliquer le caractère unique mondial du client IDs en dehors de votre région Compte AWS ou entre les régions de votre Compte AWS pays.

L'impact et la gravité de l'abandon des connexions MQTT sur votre parc d'appareils dépend de nombreux facteurs. Il s'agit des licences suivantes :

- Votre cas d'utilisation (par exemple, les données auxquelles vos appareils envoient AWS IoT, la quantité de données et la fréquence à laquelle les données sont envoyées).
- La configuration de votre client MQTT (par exemple, les paramètres de reconnexion automatique, les temporisations de back-off associées et l'utilisation de sessions persistantes MQTT).
- Contraintes liées aux ressources de l'appareil.

- La cause première des déconnexions, leur agressivité et leur persistance.

Pour éviter les conflits d'ID client et leurs impacts négatifs potentiels, assurez-vous que chaque appareil ou application mobile dispose d'une politique AWS IoT ou d'une politique IAM qui restreint le client IDs pouvant être utilisé pour les connexions MQTT au AWS IoT courtier de messages. Par exemple, vous pouvez utiliser une politique IAM pour empêcher un appareil de fermer involontairement la connexion d'un autre appareil à l'aide d'un ID client déjà utilisé. Pour de plus amples informations, veuillez consulter Autorisation.

Tous les appareils de votre parc doivent disposer d'informations d'identification avec des privilèges autorisant uniquement les actions prévues, notamment (mais sans s'y limiter) les actions AWS IoT MQTT telles que la publication de messages ou l'abonnement à des sujets ayant une portée et un contexte spécifiques. Les stratégies d'autorisation spécifiques peuvent varier selon vos cas d'utilisation. Identifiez les stratégies d'autorisation qui répondent le mieux aux exigences de votre entreprise et de sécurité.

Pour simplifier la création et la gestion des politiques d'autorisation, vous pouvez utiliser les AWS IoT Core variables de politique et les variables de la politique IAM. Les variables de la stratégie peuvent être placées dans une stratégie et lorsque celle-ci est évaluée, les variables sont remplacées par les valeurs provenant de la demande de l'appareil. Avec des variables de stratégie, vous pouvez créer une stratégie unique pour l'octroi d'autorisations à plusieurs appareils. Vous pouvez identifier les variables de politique pertinentes pour votre cas d'utilisation en fonction de la configuration de votre AWS IoT compte, du mécanisme d'authentification et du protocole réseau utilisés pour vous connecter au courtier de AWS IoT messages. Toutefois, pour écrire les meilleures stratégies d'autorisation, vous devez prendre en compte les spécificités de votre cas d'utilisation et votre modèle de menaces.

Par exemple, si vous avez enregistré vos appareils dans le AWS IoT registre, vous pouvez utiliser les variables de politique des objets dans les AWS IoT politiques pour accorder ou refuser des autorisations en fonction des propriétés des objets, telles que les noms des objets, les types d'objets et les valeurs des attributs des objets. Le nom de l'objet est obtenu à partir de l'ID client figurant dans le message de connexion MQTT envoyé lorsqu'un objet se connecte à AWS IoT. Les variables de politique des objets sont remplacées lorsqu'un objet se connecte AWS IoT via MQTT à l'aide de l'authentification mutuelle TLS ou MQTT via le WebSocket protocole à l'aide d'identités Amazon Cognito authentifiées. Vous pouvez utiliser l'AttachThingPrincipalAPI pour associer des certificats et des identités Amazon Cognito authentifiées à un objet. `iot:Connection.Thing.ThingName`est une variable de politique utile pour appliquer les restrictions d'identification des clients. L'exemple de

AWS IoT politique suivant exige que le nom d'un objet enregistré soit utilisé comme ID client pour les connexions MQTT au courtier de AWS IoT messages :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "iot:Connect",
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ]
  }
 ]
}
```

Si vous souhaitez identifier les conflits d'ID client en cours, vous pouvez activer et utiliser CloudWatch Logs for AWS IoT. Pour chaque connexion MQTT déconnectée par le courtier de AWS IoT messages en raison de conflits d'ID client, un enregistrement de journal similaire au suivant est généré :

```
{
    "timestamp": "2019-04-28 22:05:30.105",
    "logLevel": "ERROR",
    "traceId": "02a04a93-0b3a-b608-a27c-1ae8ebdb032a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "clientId01",
    "principalId": "1670fcf6de55adc1930169142405c4a2493d9eb5487127cd0091ca0193a3d3f6",
    "sourceIp": "203.0.113.1",
    "sourcePort": 21335,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID"
}
```

Vous pouvez utiliser un filtre CloudWatch Logs, par exemple `{$.reason= "DUPLICATE_CLIENT_ID" }` pour rechercher des cas de conflits d'ID client ou pour configurer des

filtres CloudWatch métriques et des CloudWatch alarmes correspondantes pour une surveillance et des rapports continus.

Vous pouvez utiliser AWS IoT Device Defender pour identifier les politiques IAM AWS IoT et les politiques trop permissives. AWS IoT Device Defender fournit également une vérification d'audit qui vous avertit si plusieurs appareils de votre parc se connectent au courtier de AWS IoT messages en utilisant le même identifiant client.

Vous pouvez utiliser AWS IoT Device Advisor pour vérifier que vos appareils peuvent se connecter de manière fiable AWS IoT Core et suivre les meilleures pratiques en matière de sécurité.

Consultez aussi

- AWS IoT Core
- Fonctions de sécuritéAWS IoT
- AWS IoT Core variables de politique
- Variables de politique IAM
- Amazon Cognito Identity
- AWS IoT Défenseur de l'appareil
- CloudWatch Logs pour AWS IoT

Veiller à la synchronisation de l'horloge de votre appareil

Il est important que l'heure soit exacte sur votre appareil. Les certificats X.509 ont une date et une heure d'expiration. L'horloge de votre appareil est utilisée pour vérifier qu'un certificat de serveur est toujours valide. Si vous construisez des appareils IoT commerciaux, n'oubliez pas que vos produits peuvent être stockés pendant de longues périodes avant d'être vendus. Les horloges en temps réel peuvent dériver pendant cette période et les batteries peuvent se décharger, par conséquent, il ne suffit pas de régler l'heure en usine.

Pour la plupart des systèmes, cela signifie que le logiciel de l’appareil doit inclure un client NTP (Network Time Protocol). L’appareil doit attendre qu'il se synchronise avec un serveur NTP avant d'essayer de se connecter à AWS IoT Core. Si ce n'est pas possible, le système doit fournir un moyen aux utilisateurs de définir l'heure de l'appareil afin que les connexions suivantes réussissent.

Une fois l'appareil synchronisé avec un serveur NTP, il peut établir une connexion avec AWS IoT Core. L’ampleur du décalage d'horloge autorisé dépend de ce que vous essayez de faire avec la connexion.

Valider le certificat de serveur

La première chose avec laquelle un appareil interagit AWS IoT est d'ouvrir une connexion sécurisée. Lorsque vous connectez votre appareil à AWS IoT, assurez-vous que vous parlez à un autre serveur AWS IoT et qu'il n'y a pas d'usurpation AWS IoT d'identité. Chacun des AWS IoT serveurs est approvisionné avec un certificat émis pour le `iot.amazonaws.com` domaine. Ce certificat a été délivré AWS IoT par une autorité de certification fiable qui a vérifié notre identité et la propriété du domaine.

L'une des premières choses à AWS IoT Core faire lorsqu'un appareil se connecte est de lui envoyer un certificat de serveur. Les appareils peuvent vérifier qu'ils sont censés se connecter à `iot.amazonaws.com` et que le serveur à l'autre extrémité de cette connexion possède un certificat provenant d'une autorité de confiance pour ce domaine.

Les certificats TLS sont au format X.509 et comprennent diverses informations, telles que le nom de l'organisation, l'emplacement, le nom de domaine et une période de validité. La période de validité est spécifiée sous la forme d'une paire de valeurs temporelles nommées `notBefore` et `notAfter`. Des services tels que AWS IoT Core l'utilisation de périodes de validité limitées (par exemple, un an) pour leurs certificats de serveur et commencent à en servir de nouveaux avant l'expiration des anciens.

Utiliser une identité unique par appareil

Utilisez une identité unique par client. Les appareils utilisent généralement des certificats client X.509. Les applications Web et mobiles utilisent Amazon Cognito Identity. Cela vous permet d'appliquer des autorisations précises à vos appareils.

Par exemple, si une application se compose d'un téléphone mobile qui reçoit des mises à jour d'état de deux objets connectés différents – une ampoule et un thermostat. L'ampoule envoie l'état de son niveau de batterie, et un thermostat envoie des messages indiquant la température.

AWS IoT authentifie les appareils individuellement et traite chaque connexion individuellement. Vous pouvez appliquer des contrôles d'accès précis grâce à des stratégies d'autorisation. Vous pouvez définir une stratégie pour le thermostat, qui l'autorise à publier dans un espace de rubrique. Vous pouvez définir une stratégie distincte pour l'ampoule, qui l'autorise à publier dans un autre espace de rubrique. Enfin, vous pouvez définir une stratégie pour l'application mobile, qui l'autorise uniquement à se connecter et à s'abonner aux rubriques du thermostat et de l'ampoule, afin de recevoir des messages de ces appareils.

Appliquez le principe du moins de privilèges et définissez les autorisations par appareil autant que possible. Tous les appareils ou utilisateurs doivent disposer d'une AWS IoT politique leur AWS IoT

permettant uniquement de se connecter avec un identifiant client connu, de publier et de s'abonner à un ensemble de sujets identifiés et fixes.

Utiliser une seconde Région AWS comme sauvegarde

Envisagez de stocker une copie de vos données en une seconde à Région AWS titre de sauvegarde. Notez que la AWS solution nommée Disaster Recovery for n' AWS IoT est plus disponible. Bien que la GitHubbibliothèque associée reste accessible, elle est AWS devenue obsolète en juillet 2023 et n'en assure plus la maintenance ni le support. Pour mettre en œuvre vos propres solutions ou pour explorer d'autres options d'assistance, rendez-vous sur Contact AWS. Si un responsable de compte AWS technique est associé à votre compte, contactez-le pour obtenir de l'aide.

Utiliser la mise en service juste à temps

La création et le provisionnement manuels de chaque appareil peuvent prendre beaucoup de temps. AWS IoT fournit un moyen de définir un modèle pour approvisionner les appareils lorsqu'ils se connectent pour la première fois à AWS IoT. Pour de plus amples informations, veuillez consulter Just-in-time approvisionnement.

Autorisations pour exécuter des tests AWS IoT Device Advisor

Le modèle de politique suivant indique les autorisations minimales et l'entité IAM requises pour exécuter les scénarios de test AWS IoT Device Advisor. Vous devrez le remplacer par le rôle *your-device-role-arn* d'appareil Amazon Resource Name (ARN) que vous avez créé conformément aux prérequis.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "your-device-role-arn",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iotdeviceadvisor.amazonaws.com"
                }
```

```
            }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke*",
                "iam:ListRoles",   // Required to list device roles in the Device
Advisor console
                "iot:Connect",
                "iot:CreateJob",
                "iot:DeleteJob",
                "iot:DescribeCertificate",
                "iot:DescribeEndpoint",
                "iotjobsdata:DescribeJobExecution",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iotjobsdata:GetPendingJobExecutions",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListCertificates",
                "iot:ListPrincipalPolicies",
                "iot:ListThingPrincipals",
                "iot:ListThings",
                "iot:Publish",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:UpdateJobExecution",
                "iot:UpdateThingShadow",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:*",
            "Resource": "*"
        }
    ]
```

```
}
```

Prévention du problème de l'adjoint confus entre services pour l'interface Device Advisor

Le problème de député confus est un problème de sécurité dans lequel une entité qui n'est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. En AWS, l'usurpation d'identité interservices peut entraîner la confusion des adjoints. L'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé et ses autorisations utilisées pour agir sur les ressources d'un autre client auxquelles on ne serait pas autorisé d'accéder autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services auprès des principaux fournisseurs de services qui ont obtenu l'accès aux ressources de votre compte.

Nous vous recommandons d'utiliser les clés de contexte `aws:SourceArn` et de condition globale `aws:SourceAccount` dans les politiques de ressources afin de limiter les autorisations à la ressource octroyées par Device Advisor à un autre service. Si vous utilisez les deux clés de contexte de condition globale, la valeur `aws:SourceAccount` et le compte de la valeur `aws:SourceArn` doit utiliser le même ID de compte lorsqu'il est utilisé dans la même déclaration de stratégie.

La valeur de `aws:SourceArn` doit être l'ARN de votre ressource de définition de suite. La ressource de définition de suite fait référence à la suite de tests que vous avez créée avec Device Advisor.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. Si vous ne connaissez pas l'ARN complet de la ressource ou si vous spécifiez plusieurs ressources, utilisez la clé de contexte de condition globale `aws:SourceArn` avec des caractères génériques (`*`) pour les parties inconnues de l'ARN. Par exemple, `arn:aws:iotdeviceadvisor:*:`*`account-id`*`:suitedefinition/*`

L'exemple suivant montre comment utiliser les clés contextuelles `aws:SourceArn` et de condition globale `aws:SourceAccount` dans Device Advisor pour éviter le problème de confusion des adjoints.

JSON

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "iotdeviceadvisor.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:iotdeviceadvisor:us-
east-1:123456789012:suitedefinition/ygp6rxa3tzvn"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

AWS formation et certification

Suivez le cours suivant pour en savoir plus sur les concepts clés de AWS IoT la sécurité : AWS IoT Security Primer.

Surveillance AWS IoT

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS IoT et des performances de vos AWS solutions.

Nous vous encourageons vivement à collecter des données de surveillance provenant de toutes les parties de votre AWS solution afin de faciliter le débogage d'une défaillance multipoint, le cas échéant. Commencez par créer un plan de surveillance qui répond aux questions suivantes. Si vous ne savez pas comment y répondre, vous pouvez continuer à activer la journalisation et établir vos lignes de base de performances.

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- A quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

L'étape suivante consiste à activer la journalisation et à établir une base de référence des AWS IoT performances normales dans votre environnement en mesurant les performances à différents moments et dans différentes conditions de charge. Pendant que vous surveillez AWS IoT, conservez les données de surveillance historiques afin de pouvoir les comparer aux données de performance actuelles. Cela vous aide à identifier les modèles de performances normaux et les anomalies de performances, et à élaborer des méthodes pour résoudre ces problèmes.

Pour établir vos performances de référence AWS IoT, vous devez d'abord surveiller ces indicateurs. Vous pouvez toujours surveiller plus de métriques ultérieurement.

- `PublishIn.Success`
- `PublishOut.Success`
- `Subscribe.Success`
- `Ping.Success`
- `Connect.Success`
- `GetThingShadow.Accepted`

- `UpdateThingShadow.Accepted`
- `DeleteThingShadow.Accepted`
- `RulesExecuted`

Les rubriques de cette section peuvent vous aider à démarrer la journalisation et la surveillance de AWS IoT.

Rubriques

- Configuration de la AWS IoT journalisation
- Surveillez les AWS IoT alarmes et les métriques à l'aide d'Amazon CloudWatch
- Surveiller AWS IoT à l'aide CloudWatch des journaux
- Téléchargez les journaux côté appareil sur Amazon CloudWatch
- Enregistrement des appels AWS IoT d'API à l'aide de AWS CloudTrail

Configuration de la AWS IoT journalisation

Vous devez activer la journalisation à l'aide de la AWS IoT console, de la CLI ou de l'API avant de pouvoir surveiller et consigner AWS IoT l'activité.

Vous pouvez activer la journalisation pour tous les groupes d' AWS IoT objets ou uniquement pour certains d'entre eux. Vous pouvez configurer la AWS IoT journalisation à l'aide de la AWS IoT console, de la CLI ou de l'API ; toutefois, vous devez utiliser la CLI ou l'API pour configurer la journalisation pour des groupes d'objets spécifiques.

Lorsque vous réfléchissez à la configuration de votre AWS IoT journalisation, la configuration de journalisation par défaut détermine la manière dont AWS IoT l'activité sera enregistrée, sauf indication contraire. À partir de là, vous pouvez obtenir des journaux détaillés avec un niveau de journal par défaut de `INFO` ou `DEBUG`. Après avoir examiné les journaux initiaux, vous pouvez modifier le niveau de journal par défaut à un niveau moins détaillé tel que `WARN` ou `ERROR`, et définir un niveau de journal plus détaillé spécifique à la ressource sur les ressources qui pourraient nécessiter plus d'attention. Les niveaux de journal peuvent être modifiés quand vous le souhaitez.

Cette rubrique traite de la connexion côté cloud. AWS IoT Pour plus d'informations sur la journalisation et la surveillance côté appareil, voir Télécharger les journaux côté appareil vers. CloudWatch

Pour plus d'informations sur la journalisation et AWS IoT Greengrass la surveillance, consultez la section Connexion et surveillance AWS IoT Greengrass. Au 30 juin 2023, le logiciel AWS IoT Greengrass Core a migré vers AWS IoT Greengrass Version 2.

Configurer le rôle et la stratégie de journalisation

Avant de pouvoir activer la connexion AWS IoT, vous devez créer un rôle IAM et une politique AWS autorisant à surveiller l' AWS IoT activité en votre nom. Vous pouvez également générer un rôle IAM avec les politiques nécessaires dans la section Logs de la AWS IoT console.

> **Note**
>
> Avant d'activer la AWS IoT journalisation, assurez-vous de bien comprendre les autorisations d'accès aux CloudWatch journaux. Les utilisateurs ayant accès aux CloudWatch journaux peuvent consulter les informations de débogage de vos appareils. Pour plus d'informations, consultez Authentification et contrôle d'accès pour Amazon CloudWatch Logs.
> Si vous vous attendez à des modèles de trafic élevés en AWS IoT Core raison des tests de charge, pensez à désactiver la journalisation de l'IoT pour éviter les ralentissements. Si un trafic élevé est détecté, notre service peut désactiver la connexion à votre compte.

Vous trouverez ci-dessous comment créer un rôle et une politique de journalisation pour les AWS IoT Core ressources.

Création d'un rôle de journalisation

Pour créer un rôle de journalisation, ouvrez le hub Roles de la console IAM et choisissez Create role.

1. Sous Sélectionner une entité approuvée, choisissez AWS Service . Choisissez ensuite IoT sous Cas d'utilisation. Si vous ne voyez pas IoT, saisissez et recherchez IoT dans le menu déroulant Cas d'utilisation pour d'autres AWS services :. Sélectionnez Suivant.
2. Sur la page Ajouter des autorisations, vous verrez les politiques automatiquement associées au rôle de service. Choisissez Suivant.
3. Sur la page Name, review, and Create (Créer un rôle et vérifier), saisissez un Role name (Nom du rôle) etRole description (Description de Role) pour le role, puis Create role(Créer un rôle).
4. Dans la liste des rôles, recherchez le rôle que vous avez créé, ouvrez-le et copiez l'ARN du rôle (*`logging-role-arn`*) à utiliser lorsque vous le souhaitezConfigurez la journalisation par défaut dans AWS IoT (console).

Stratégie de rôle de journalisation

Les documents de politique suivants fournissent la politique de rôle et la politique de confiance qui AWS IoT permettent de soumettre des entrées de journal CloudWatch en votre nom. Si vous êtes également autorisé AWS IoT Core for LoRaWAN à soumettre des entrées de journal, vous verrez un document de politique créé pour vous qui enregistre les deux activités.

> **Note**
>
> Ces documents ont été créés pour vous lorsque vous avez créé le rôle de journalisation. Les documents contiennent des variables *${partition},${region}*,*${accountId}*, et que vous devez remplacer par vos valeurs.

Politique de rôle :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream",
    "logs:PutLogEvents",
    "logs:PutMetricFilter",
    "logs:PutRetentionPolicy",
    "iot:GetLoggingOptions",
    "iot:SetLoggingOptions",
    "iot:SetV2LoggingOptions",
    "iot:GetV2LoggingOptions",
    "iot:SetV2LoggingLevel",
    "iot:ListV2LoggingLevels",
    "iot:DeleteV2LoggingLevel"
   ],
   "Resource": [
    "arn:${partition}:logs:${region}:${accountId}:log-group:AWSIotLogsV2:*"
   ]
  }
```

```
 ]
}
```

Politique de confiance permettant de ne consigner que AWS IoT Core l'activité :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

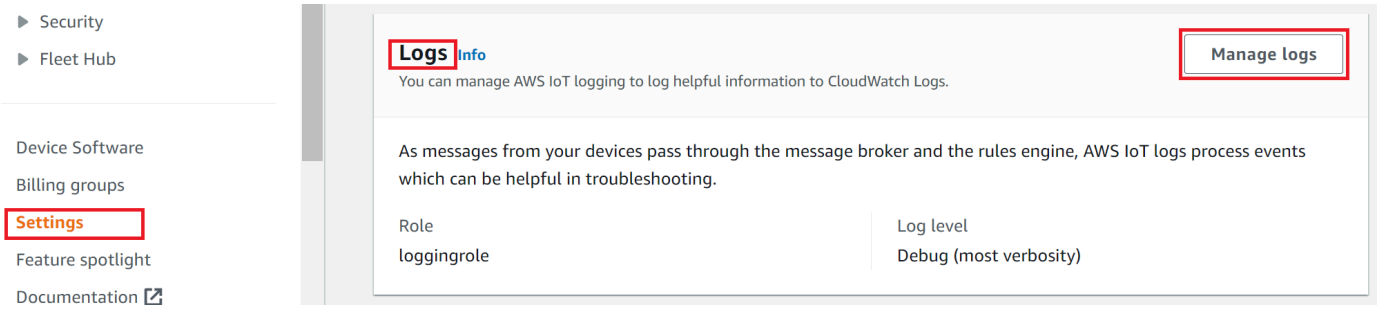
Configurez la journalisation par défaut dans AWS IoT (console)

Cette section décrit comment utiliser la AWS IoT console pour configurer la journalisation pour tous AWS IoT. Pour configurer la journalisation uniquement pour des groupes de choses spécifiques, vous devez utiliser l'interface de ligne de commande ou l'API. Pour de plus amples informations sur la configuration de la journalisation pour des groupes de choses spécifiques, veuillez consulter Configurer la connexion spécifique aux ressources AWS IoT (interface de ligne de commande).

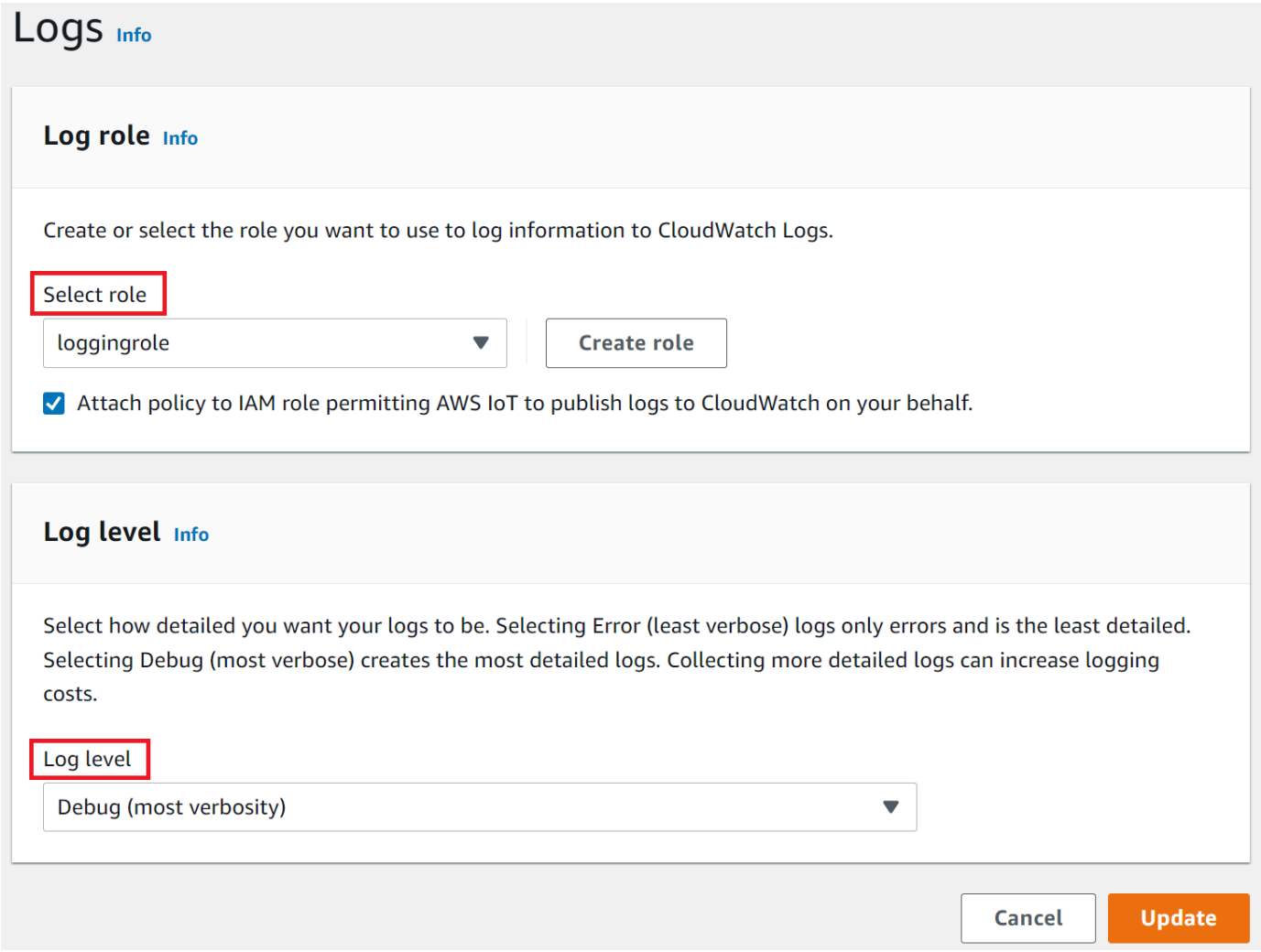
Pour utiliser la AWS IoT console afin de configurer la journalisation par défaut pour tous AWS IoT

1. Connectez-vous à la AWS IoT console. Pour de plus amples informations, veuillez consulter Ouvrez la AWS IoT console.
2. Dans le panneau de navigation de gauche, choisissez Paramètres. Dans la section Journaux de la page Paramètres, choisissez Gérer les journaux.

   La page Journaux affiche le rôle de journalisation et le niveau de verbosité utilisés par tous les AWS IoT.



3. sur la page Journaux choisissez Sélectionner le rôle pour spécifier un rôle que vous avez créé dans Création d'un rôle de journalisation, ou Créer un rôle pour créer un rôle à utiliser pour la journalisation.



4. Choisissez le niveau de journal qui décrit le niveau de détail des entrées de journal que vous souhaitez voir apparaître dans les CloudWatch journaux.
5. Choisissez Mettre à jour pour enregistrer vos modifications.

Une fois que vous avez activé la journalisation, visitez Afficher AWS IoT les journaux dans la CloudWatch console pour en savoir plus sur l'affichage des entrées du journal.

Configurer la connexion par défaut AWS IoT (CLI)

Cette section décrit comment configurer la journalisation globale pour à l'aide AWS IoT de la CLI.

> **Note**
>
> Vous avez besoin du nom de ressource Amazon (ARN) du rôle que vous souhaitez utiliser. Si vous devez créer un rôle à utiliser pour la journalisation, veuillez consulter Création d'un rôle de journalisation avant de continuer.
> Le principal utilisé pour appeler l'API doit avoir Transmission des autorisations de rôle pour votre rôle de journalisation.

Vous pouvez également exécuter cette procédure avec l'API en utilisant les méthodes de l' AWS API qui correspondent aux commandes CLI présentées ici.

Pour utiliser la CLI afin de configurer la journalisation par défaut pour AWS IoT

1. Utilisez la commande set-v2-logging-options pour définir les options de journalisation de votre compte.

   ```
   aws iot set-v2-logging-options \
       --role-arn logging-role-arn \
       --default-log-level log-level
   ```

   où :

   --role-arn

   Le rôle ARN qui accorde AWS IoT l'autorisation d'écrire dans vos CloudWatch journaux dans Logs.

   --default-log-level

   Le niveau de journalisation à utiliser. Les valeurs valables sont `ERROR`, `WARN`, `INFO`, `DEBUG` ou `DISABLED`

--no-disable-all-logs

Paramètre facultatif qui active tous les AWS IoT enregistrements. Utilisez ce paramètre pour activer la journalisation lorsqu'elle est désactivée.

--disable-all-logs

Paramètre facultatif qui désactive toute AWS IoT journalisation. Utilisez ce paramètre pour désactiver la journalisation lorsqu'elle est activée.

2. Utilisez la commande get-v2-logging-options pour obtenir vos options de journalisation actuelles.

```
aws iot get-v2-logging-options
```

Une fois que vous avez activé la journalisation, visitez Afficher AWS IoT les journaux dans la CloudWatch console pour en savoir plus sur l'affichage des entrées du journal.

> **Note**
>
> AWS IoT continue de prendre en charge les anciennes commandes (set-logging-optionsetget-logging-options) permettant de configurer et d'obtenir une connexion globale sur votre compte. Lorsque ces commandes sont utilisées, les journaux obtenus contiennent du texte brut et non des charges de travail JSON et la latence de journalisation est généralement plus élevée. Aucune amélioration supplémentaire ne sera apportée à l'implémentation de ces anciennes commandes. Nous vous recommandons d'utiliser les versions « v2 » pour configurer vos options de journalisation et, si possible, de modifier toutes les applications existantes qui utilisent les anciennes versions.

Configurer la connexion spécifique aux ressources AWS IoT (interface de ligne de commande)

Cette section décrit comment configurer la journalisation spécifique aux ressources à l'aide de AWS IoT la CLI. La journalisation spécifique à la ressource vous permet de spécifier un niveau de journalisation pour un groupe d'objetsspécifique.

Les groupes d'objets peuvent contenir d'autres groupes d'objets pour créer une relation hiérarchique. Cette procédure décrit comment configurer la journalisation d'un seul groupe d'objets. Vous pouvez appliquer cette procédure au groupe d'objets parent dans une hiérarchie pour configurer

la journalisation de tous les groupes d'objets de la hiérarchie. Vous pouvez également appliquer cette procédure à un groupe d'objets enfant pour remplacer la configuration de journalisation de son parent.

Un objet peut être membre d'un groupe d'objets. Cette appartenance permet à l'objet d'hériter des configurations, des politiques et des paramètres appliqués au groupe d'objets. Les groupes d'objets sont utilisés pour gérer et appliquer des paramètres à plusieurs éléments collectivement, plutôt que de traiter chaque élément individuellement. Lorsque votre ID client correspond au nom de l'objet, la session client est AWS IoT Core automatiquement associée à la ressource d'objet correspondante. Cela permet à la session client d'hériter des configurations et des paramètres appliqués aux groupes d'objets auxquels appartient l'objet, y compris les niveaux de journalisation. Si votre identifiant client ne correspond pas au nom de l'objet, vous pouvez activer la pièce jointe exclusive pour établir l'association. Pour de plus amples informations, veuillez consulter ???.

Outre les groupes d'objets, vous pouvez également enregistrer des cibles telles que l'ID client, l'IP source et l'ID principal d'un appareil.

> **Note**
>
> Vous avez besoin du nom de ressource Amazon (ARN) du rôle que vous souhaitez utiliser. Si vous devez créer un rôle à utiliser pour la journalisation, veuillez consulter Création d'un rôle de journalisation avant de continuer.
> Le principal utilisé pour appeler l'API doit avoir Transmission des autorisations de rôle pour votre rôle de journalisation.

Vous pouvez également exécuter cette procédure avec l'API en utilisant les méthodes de l' AWS API qui correspondent aux commandes CLI présentées ici.

Pour utiliser la CLI afin de configurer la journalisation spécifique aux ressources pour AWS IoT

1. Utilisez la commande set-v2-logging-options pour définir les options de journalisation de votre compte.

   ```
   aws iot set-v2-logging-options \
       --role-arn logging-role-arn \
       --default-log-level log-level
   ```

   où :

--role-arn

Le rôle ARN qui accorde AWS IoT l'autorisation d'écrire dans vos CloudWatch journaux dans Logs.

--default-log-level

Le niveau de journalisation à utiliser. Les valeurs valables sont `ERROR`, `WARN`, `INFO`, `DEBUG` ou `DISABLED`

--no-disable-all-logs

Paramètre facultatif qui active tous les AWS IoT enregistrements. Utilisez ce paramètre pour activer la journalisation lorsqu'elle est désactivée.

--disable-all-logs

Paramètre facultatif qui désactive toute AWS IoT journalisation. Utilisez ce paramètre pour désactiver la journalisation lorsqu'elle est activée.

2. Utilisez la commande set-v2-logging-level pour configurer la journalisation spécifique à une ressource pour un groupe de choses.

```
aws iot set-v2-logging-level \
              --log-target targetType=THING_GROUP,targetName=thing_group_name \
              --log-level log_level
```

--log-target

Type et nom de la ressource pour laquelle vous configurez la journalisation. La `target_type` valeur doit être l'une des suivantes : `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` |`PRINCIPAL_ID`. La valeur du paramètre log-target peut être du texte, comme indiqué dans l'exemple de commande précédent, ou une chaîne JSON, telle que l'exemple suivant.

```
aws iot set-v2-logging-level \
              --log-target '{"targetType": "THING_GROUP","targetName":
 "thing_group_name"}' \
              --log-level log_level
```

--log-level

Le niveau de journalisation utilisé lors de la génération de journaux pour la ressource spécifiée. Les valeurs valides sont : DEBUG, INFO, ERROR, WARN et DISABLED.

```
aws iot set-v2-logging-level \
              --log-target targetType=CLIENT_ID,targetName=ClientId1 \
              --log-level DEBUG
```

3. Utilisez la commande list-v2-logging-levels pour répertorier les niveaux de journalisation actuellement configurés.

```
aws iot list-v2-logging-levels
```

4. Utilisez la commande delete-v2-logging-level pour supprimer un niveau de journalisation spécifique à la ressource, tel que les exemples suivants.

```
aws iot delete-v2-logging-level \
              --target-type "THING_GROUP" \
              --target-name "thing_group_name"
```

```
aws iot delete-v2-logging-level \
              --target-type=CLIENT_ID
              --target-name=ClientId1
```

--targetType

La `target_type` valeur doit être l'une des suivantes : `THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` |`PRINCIPAL_ID`.

--targetName

Nom du groupe d'objets pour lequel le niveau de journalisation doit être supprimé.

Une fois que vous avez activé la journalisation, visitez Afficher AWS IoT les journaux dans la CloudWatch console pour en savoir plus sur l'affichage des entrées du journal.

Niveaux de journalisation.

Ces niveaux de journal déterminent les événements qui sont consignés et s'appliquent aux niveaux de journal par défaut et spécifiques aux ressources.

ERROR

Toute erreur qui entraîne l'échec d'une opération.

Les journaux contiennent uniquement des informations ERROR.

WARN

Tout ce qui peut éventuellement entraîner des incohérences dans le système, mais qui n'entraîne pas nécessairement l'échec de l'opération.

Les journaux contiennent des informations ERROR et WARN.

INFO

Informations générales sur le flux des objets.

Les journaux contiennent des informations INFO, ERROR et WARN.

DEBUG

Informations qui peuvent être utiles lors du débogage d'un problème.

Les journaux contiennent des informations DEBUG, INFO, ERROR et WARN.

DISABLED

Toute la journalisation est désactivée.

Surveillez les AWS IoT alarmes et les métriques à l'aide d'Amazon CloudWatch

Vous pouvez surveiller AWS IoT l'utilisation CloudWatch, qui collecte et traite les données brutes sous forme de métriques lisibles AWS IoT en temps quasi réel. Ces statistiques sont enregistrées pour une durée de deux semaines et, par conséquent, vous pouvez accéder aux informations historiques et acquérir un meilleur point de vue de la façon dont votre service ou application web

s'exécute. Par défaut, les données AWS IoT métriques sont envoyées automatiquement CloudWatch à des intervalles d'une minute. Pour plus d'informations, consultez Que sont Amazon CloudWatch, Amazon CloudWatch Events et Amazon CloudWatch Logs ? dans le guide de CloudWatch l'utilisateur Amazon.

Utilisation de AWS IoT métriques

Les statistiques rapportées par AWS IoT fournissent des informations que vous pouvez analyser de différentes manières. Les cas d'utilisation suivants sont basés sur un scénario où vous avez dix objets qui se connectent à Internet une fois par jour. Chaque jour :

- Dix objets se connectent AWS IoT à peu près en même temps.
- Chaque objet s'abonne à un filtre de rubrique, puis attend une heure avant de déconnecter. Au cours de cette période, les objets communiquent entre eux pour en savoir plus sur l'état du monde.
- Chaque objet publie sa perception, d'après les données qu'il vient de détecter avec `UpdateThingShadow`.
- Chaque objet se déconnecte de. AWS IoT

Pour vous aider à démarrer, ces rubriques explorent certaines des questions que vous pourriez avoir.

- Comment puis-je être informé si mes objets ne se connectent pas chaque jour ?
- Comment puis-je être informé si mes objets ne publient pas de données chaque jour ?
- Comment puis-je être informé si les mises à jour de mon thing shadow sont rejetées chaque jour ?
- Comment créer une CloudWatch alarme pour Jobs ?

En savoir plus sur les CloudWatch alarmes et les métriques

- Création d' CloudWatch alarmes à surveiller AWS IoT
- AWS IoT métriques et dimensions

Création d' CloudWatch alarmes à surveiller AWS IoT

Vous pouvez créer une CloudWatch alarme qui envoie un message Amazon SNS lorsque l'alarme change d'état. Une alarme surveille une métrique sur la période que vous spécifiez. Lorsque la valeur de la métrique dépasse un seuil donné sur un certain nombre de périodes, une ou plusieurs actions sont effectuées. L'action est une notification envoyée à une rubrique d’Amazon SNS ou à

une politique AutoScaling. Les alarmes déclenchent des actions uniquement pour les changements d'état prolongés. CloudWatch les alarmes ne déclenchent pas d'actions simplement parce qu'elles se trouvent dans un état particulier ; cet état doit avoir changé et être maintenu pendant un certain nombre de périodes.

Les rubriques suivantes décrivent quelques exemples d'utilisation d'alarmes CloudWatch.

- Comment puis-je être informé si mes objets ne se connectent pas chaque jour ?
- Comment puis-je être informé si mes objets ne publient pas de données chaque jour ?
- Comment puis-je être informé si les mises à jour du shadow de mon objet sont rejetées chaque jour ?
- Comment créer une CloudWatch alarme pour les offres d'emploi ?

Vous pouvez voir tous les indicateurs que les CloudWatch alarmes peuvent surveillerAWS IoT métriques et dimensions.

Comment puis-je être informé si mes objets ne se connectent pas chaque jour ?

1. Création d'une rubrique Amazon SNS nommée `things-not-connecting-successfully`, et enregistrement de son Nom de Ressource Amazon (ARN). Cette procédure fera référence à l'ARN de votre rubrique en tant que *`sns-topic-arn`*.

   Pour de plus amples informations sur la création d'une notification, veuillez consulter Démarrer avec avec Amazon SNS.
2. Créez l'alerte.

```
aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
    --alarm-description "Alarm when my Things don't connect successfully" \
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Testez l'alarme.

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Vérifiez que l'alarme s'affiche dans votre console CloudWatch .

Comment puis-je être informé si mes objets ne publient pas de données chaque jour ?

1. Créez une rubrique Amazon SNS nommée `things-not-publishing-data`, et enregistrez son Nom de Ressource Amazon (ARN). Cette procédure fera référence à l'ARN de votre rubrique en tant que *`sns-topic-arn`*.

   Pour de plus amples informations sur la création d'une notification, veuillez consulter Démarrer avec avec Amazon SNS.

2. Créez l'alerte.

```
aws cloudwatch put-metric-alarm \
    --alarm-name PublishInSuccessAlarm\
    --alarm-description "Alarm when my Things don't publish their data \
    --namespace AWS/IoT \
    --metric-name PublishIn.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Testez l'alarme.

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. Vérifiez que l'alarme s'affiche dans votre console CloudWatch .

Comment puis-je être informé si les mises à jour du shadow de mon objet sont rejetées chaque jour ?

1. Créez une rubrique Amazon SNS nommée `things-shadow-updates-rejected`, et enregistrez son Nom de Ressource Amazon (ARN). Cette procédure fera référence à l'ARN de votre rubrique en tant que *`sns-topic-arn`*.

   Pour de plus amples informations sur la création d'une notification, veuillez consulter Démarrer avec avec Amazon SNS.

2. Créez l'alerte.

```
aws cloudwatch put-metric-alarm \
    --alarm-name UpdateThingShadowSuccessAlarm \
    --alarm-description "Alarm when my Things Shadow updates are getting rejected"
 \
    --namespace AWS/IoT \
    --metric-name UpdateThingShadow.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. Testez l'alarme.

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value ALARM
```

4. Vérifiez que l'alarme s'affiche dans votre console CloudWatch .

Comment créer une CloudWatch alarme pour les offres d'emploi ?

Le service Jobs fournit des CloudWatch statistiques qui vous permettent de suivre vos offres d'emploi. Vous pouvez créer des alarmes CloudWatch pour surveiller les Métriques de tâches.

La commande suivante crée une CloudWatch alarme pour surveiller le nombre total d'exécutions de tâches ayant échoué pour Job *SampleOTAJob* et vous avertit lorsque plus de 20 exécutions de tâches ont échoué. L'alarme surveille la métrique Jobs `FailedJobExecutionTotalCount` en vérifiant la valeur signalée toutes les 300 secondes. Il est activé lorsqu'une seule valeur signalée est supérieure à 20, ce qui signifie qu'il y a eu plus de 20 exécutions de travail ayant échoué depuis le début de la tâche. Lorsque l'alarme s'éteint, elle envoie une notification à la rubrique Amazon SNS fournie.

```
aws cloudwatch put-metric-alarm \
    --alarm-name TotalFailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when total number of failed job execution exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionTotalCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 20 \
    --comparison-operator GreaterThanThreshold \
    --period 300 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions
```

La commande suivante crée une CloudWatch alarme pour surveiller le nombre d'échecs d'exécution de tâches pour Job *SampleOTAJob* au cours d'une période donnée. Vous êtes ensuite averti quand plus de cinq exécutions de travail ont échoué au cours de cette période. L'alarme surveille la métrique Jobs `FailedJobExecutionCount` en vérifiant la valeur signalée toutes les 3600 secondes. Il est activé lorsqu'une seule valeur signalée est supérieure à 5, ce qui signifie qu'il y a eu plus de 5 exécutions de travail ayant échoué au cours de la dernière heure. Lorsque l'alarme s'éteint, elle envoie une notification à la rubrique Amazon SNS fournie.

```
aws cloudwatch put-metric-alarm \
    --alarm-name FailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when number of failed job execution per hour exceeds the
 threshold for SampleOTAJob" \
```

```
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 5 \
    --comparison-operator GreaterThanThreshold \
    --period 3600 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-
many-failed-job-ececutions-per-hour
```

AWS IoT métriques et dimensions

Lorsque vous interagissez avec AWS IoT, le service envoie des métriques et des dimensions à CloudWatch chaque minute. Vous pouvez utiliser AWS IoT, utiliser la CloudWatch console ou AWS CLI consulter ces métriques.

Pour consulter les métriques à l'aide de la CloudWatch console, CloudWatch ouvrez-la. Dans le panneau de navigation, choisissez Metrics (Métriques), puis choisissez All metrics (Toutes les métriques). Dans l'onglet Parcourir, recherchez AWS IoT pour afficher la liste des mesures. Les métriques sont d'abord regroupées par espace de noms de service, puis par les différentes combinaisons de dimension au sein de chaque espace de noms.

Pour afficher les métriques à l'aide de AWS CLI, exécutez la commande suivante.

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch affiche les groupes de mesures suivants pour AWS IoT :

- AWS IoT métriques
- AWS IoT Core métriques du fournisseur d'informations d'identification
- Métriques d'authentification
- Métriques d'agrafage OCSP du certificat de serveur
- Métriques de règle
- Métriques d'action de règle
- Métriques spécifiques à l'action HTTP
- Métriques d'agent de messages

- Métriques de shadow d'appareil
- Métriques de tâches
- Métriques d'audit Device Defender
- Métriques de détection Device Defender
- Métriques de mise en service d'appareils
- LoRaMétriques WAN
- Métriques d'indexation de la flotte
- Dimensions pour les métriques

AWS IoT métriques

Métrique	Description
`AddThingToDynamicThingGroupsFailed`	Nombre d'événements d'échec associés à l'ajout d'un objet à un groupe d'objets dynamiques. La dimension `DynamicThingGroupName` contient le nom des groupes dynamiques qui n'ont pas pu ajouter des objets.
`NumLogBatchesFailedToPublishThrottled`	Le lot singulier d'événements de journaux qui ne s'est pas publié en raison d'erreurs de limitation.
`NumLogEventsFailedToPublishThrottled`	Le nombre d'événements de journaux au sein du lot qui ne s'est pas publié en raison d'erreurs de limitation.

AWS IoT Core métriques du fournisseur d'informations d'identification

Métrique	Description
`CredentialExchangeSuccess`	Le nombre de `AssumeRoleWithCertificate` requêtes réussies adressées au AWS IoT Core fournisseur d'informations d'identification.

Métriques d’authentification

> **Note**
>
> Les métriques d'authentification sont affichées dans la CloudWatch console sous Protocol Metrics.

Métrique	Description
`Connection.AuthNError`	Nombre de tentatives de connexion AWS IoT Core rejetées en raison d'échecs d'authentification. Cette métrique ne prend en compte que les connexion s qui envoient une chaîne d'indication de nom de serveur (SNI) correspondant à un point de terminais on de votre Compte AWS. Cette métrique inclut les tentatives de connexion provenant de sources externes telles que les outils d'analyse Internet ou les activités de sondage. La `Protocol` dimension contient le protocole utilisé pour envoyer la tentative de connexion.

Métriques d'agrafage OCSP du certificat de serveur

Métrique	Description
Récupération de OCSPStaple données : succès	La réponse OCSP a été reçue et traitée avec succès. Cette réponse sera incluse lors de la prise de contact TLS pour le domaine configuré. La `DomainCon figurationName` dimension contient le nom du domaine configuré pour lequel l'agrafage OCSP des certificats de serveur est activé.

Métriques de règle

Métrique	Description
`ParseError`	Nombre d'erreurs d'analyse JSON s'étant produites dans des messages publiés dans une rubrique dans laquelle une règle écoute. La dimension `RuleName` contient le nom de la règle.
`RuleExecutionThrottled`	Le moteur de règles limite le nombre de messages en raison d'un comportement malveillant ou parce que le nombre de messages dépasse la limite du moteur de règles. La dimension `RuleName` contient le nom de la règle à déclencher.
`RuleNotFound`	La règle à déclencher est introuvable. La dimension `RuleName` contient le nom de la règle.
`RulesExecuted`	Le nombre de AWS IoT règles exécutées.
`TopicMatch`	Nombre de messages entrants publiés dans une rubrique dans laquelle une règle écoute. La dimension `RuleName` contient le nom de la règle.

Métriques d'action de règle

Métrique	Description
`Failure`	Nombre d'appels d'action de règle en échec. La dimension `RuleName` contient le nom de la règle qui spécifie l'action. La dimension `ActionType` contient le type d'action ayant été appelé.
`Success`	Nombre d'appels d'action de règle réussis. La dimension `RuleName` contient le nom de la règle qui spécifie l'action. La dimension `ActionType` contient le type d'action ayant été appelé.

Métrique	Description
`ErrorActionFailure`	Nombre d'actions d'erreur ayant échoué. La dimension `RuleName` contient le nom de la règle qui spécifie l'action. La dimension `ActionType` contient le type d'action ayant été appelé.
`ErrorActionSuccess`	Le nombre d'actions d'erreur réussies. La dimension `RuleName` contient le nom de la règle qui spécifie l'action. La dimension `ActionType` contient le type d'action ayant été appelé.

Métriques spécifiques à l'action HTTP

Métrique	Description
`HttpCode_Other`	Généré si le code d'état de la réponse provenant du Web en aval n' service/application est pas 2xx, 4xx ou 5xx.
`HttpCode_4XX`	Généré si le code d'état de la réponse provenant du Web en aval service/application est compris entre 400 et 499.
`HttpCode_5XX`	Généré si le code d'état de la réponse provenant du Web en aval service/application est compris entre 500 et 599.
`HttpInvalidUrl`	Généré si une URL de point de terminaison, après remplacement des modèles de substitution, ne commence pas par `https://`.
`HttpRequestTimeout`	Généré si le site Web en aval service/application ne renvoie pas de réponse dans le délai d'expiration de la demande. Pour de plus amples informations, veuillez consulter Quotas de service.

Métrique	Description
`HttpUnknownHost`	Généré si l'URL est valide, mais que le service n'existe pas ou est inaccessible.

Métriques d'agent de messages

> **Note**
>
> Les métriques du courtier de messages sont affichées dans la CloudWatch console sous Protocol Metrics.

Métrique	Description
`Connect.AuthError`	Nombre de demandes de connexion n'ayant pas pu être autorisées par l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `CONNECT`.
`Connect.ClientError`	Le nombre de demandes de connexion rejetées, car le message MQTT ne respectait pas les exigences définies dans AWS IoTQuotas . La dimension `Protocol` contient le protocole utilisé pour envoyer le message `CONNECT`.
`Connect.ClientIDThrottle`	Nombre de demandes de connexion limitées car le client a dépassé le taux de demandes de connexion autorisé pour un ID client spécifique. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `CONNECT`.
`Connect.ServerError`	Nombre de demandes de connexion ayant échoué à cause d'une erreur interne. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `CONNECT`.

Métrique	Description
`Connect.Success`	Nombre de connexions réussies à l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `CONNECT`.
`Connect.Throttle`	Nombre de demandes de connexion ayant été limitées car le compte dépassait le taux de demandes de connexion autorisé. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `CONNECT`.
`Ping.Success`	Nombre de messages ping reçus par l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message ping.
`PublishIn.AuthError`	Nombre de demandes de publication que l'agent de messages n'a pas pu autoriser. La dimension `Protocol` contient le protocole utilisé pour publier le message. HTTP Publish ne prend pas en charge cette métrique.
`PublishIn.ClientError`	Le nombre de demandes de publication rejetées par l'agent de messages, car le message ne respectait pas les exigences définies dans les AWS IoTQuotas . La dimension `Protocol` contient le protocole utilisé pour publier le message.
`PublishIn.ServerError`	Nombre de demandes de publication que l'agent de messages n'a pas pu traiter à cause d'une erreur interne. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`PublishIn.Success`	Nombre de demandes de publication traitées avec succès par l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.

Métrique	Description
`PublishIn.Throttle`	Nombre de demandes de publication ayant été limitées car le client dépassait le taux de messages entrants autorisé. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`. HTTP Publish ne prend pas en charge cette métrique.
`PublishOut.AuthError`	Nombre de demandes de publication effectuées par l'agent de messages n'ayant pas pu être autorisée s par AWS IoT. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`PublishOut.ClientError`	Le nombre de demandes de publication effectuées par l'agent de messages qui ont été rejetées, car le message ne respectait pas les exigences définies dans AWS IoTQuotas . La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`PublishOut.Success`	Nombre de demandes de publication effectuées avec succès par l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`PublishOut.Throttle`	Nombre de demandes de publication qui étaient limitées parce que le client dépassait le taux de messages entrants autorisé. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`PublishRetained.AuthError`	Nombre de demandes de publication avec `RETAIN` l'indicateur activé que l'agent de messages n'a pas pu autoriser. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.

Métrique	Description
`PublishRetained.ServerError`	Nombre de demandes de publication retenues que l'agent de messages n'a pas pu traiter à cause d'une erreur interne. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`PublishRetained.Success`	Nombre de demandes de publication avec `RETAIN` l'indicateur activé traitées avec succès par l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`PublishRetained.Throttle`	Nombre de demandes de publication avec `RETAIN` l'indicateur activé ayant été limitées car le client dépassait le taux de messages entrants autorisé. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `PUBLISH`.
`Queued.Success`	Nombre de messages stockés qui ont été traités avec succès par l'agent de messages pour les clients déconnectés de leur session permanent e. Les messages dont le QoS est égal à 1 sont stockés pendant qu'un client disposant d'une session permanente est déconnecté.
`Queued.Throttle`	Le nombre de messages qui n'ont pas pu être stockés et qui ont été limités alors que les clients ayant des sessions persistantes étaient déconnect és. Cela se produit lorsque les clients dépassent la limite de messages en file d'attente par seconde et par compte. Les messages dont le QoS est égal à 1 sont stockés pendant qu'un client disposant d'une session permanente est déconnecté.

Métrique	Description
`Queued.ServerError`	Le nombre de messages qui n'ont pas été stockés pour une session persistante en raison d'une erreur interne. Lorsque les clients disposant d'une session permanente sont déconnectés, les messages dont la qualité de service (QoS) est égale à 1 sont stockés.
`Subscribe.AuthError`	Nombre de demandes d'abonnement adressées par un client et n'ayant pas pu être autorisées. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `SUBSCRIBE` .
`Subscribe.ClientError`	Le nombre de demandes d'abonnement rejetées, car le message `SUBSCRIBE` ne respectait pas les exigences définies dans AWS IoTQuotas . La dimension `Protocol` contient le protocole utilisé pour envoyer le message `SUBSCRIBE` .
`Subscribe.ServerError`	Nombre de demandes d'abonnement ayant été rejetées à cause d'une erreur interne. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `SUBSCRIBE` .
`Subscribe.Success`	Nombre de demandes d'abonnement traitées avec succès par l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `SUBSCRIBE` .

Métrique	Description
`Subscribe.Throttle`	Le nombre de demandes d'abonnement qui ont été limitées parce que les limites de taux de demandes d'abonnement autorisées ont été dépassées pour votre. Compte AWS Ces limites incluent les abonnements par seconde par compte, les abonnements par compte et les abonnements par connexion décrits dans le courtier de AWS IoT Core messages et les limites et quotas du protocole. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `SUBSCRIBE` .
`Throttle.Exceeded`	Cette métrique s'affiche CloudWatch lorsqu'un client MQTT est limité en nombre de paquets par seconde et par niveau de connexion. Cette métrique ne s'applique pas aux connexions HTTP.
`Unsubscribe.ClientError`	Le nombre de demandes de désabonnement rejetées, car le message `UNSUBSCRIBE` ne respectait pas les exigences définies dans AWS IoTQuotas . La dimension `Protocol` contient le protocole utilisé pour envoyer le message `UNSUBSCRIBE` .
`Unsubscribe.ServerError`	Nombre de demandes d'annulation d'abonnement ayant été rejetées à cause d'une erreur interne. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `UNSUBSCRIBE` .
`Unsubscribe.Success`	Nombre de demandes d'annulation d'abonnement traitées avec succès par l'agent de messages. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `UNSUBSCRIBE` .

Métrique	Description
`Unsubscribe.Throttle`	Nombre de demandes d'annulation d'abonnement ayant été rejetées car le client dépassait le taux de demandes d'annulation d'abonnement autorisé. La dimension `Protocol` contient le protocole utilisé pour envoyer le message `UNSUBSCRIBE` .

Métriques de shadow d'appareil

> **Note**
>
> Les métriques fantômes de l'appareil sont affichées dans la CloudWatch console sous Protocol Metrics.

Métrique	Description
`DeleteThingShadow.Accepted`	Nombre de demandes `DeleteThingShadow` traitées avec succès. La dimension `Protocol` contient le protocole utilisé pour effectuer la demande.
`GetThingShadow.Accepted`	Nombre de demandes `GetThingShadow` traitées avec succès. La dimension `Protocol` contient le protocole utilisé pour effectuer la demande.
`ListThingShadow.Accepted`	Nombre de demandes `ListThingShadow` traitées avec succès. La dimension `Protocol` contient le protocole utilisé pour effectuer la demande.
`UpdateThingShadow.Accepted`	Nombre de demandes `UpdateThingShadow` traitées avec succès. La dimension `Protocol` contient le protocole utilisé pour effectuer la demande.

Métriques de tâches

Métrique	Description
`CanceledJobExecutionCount`	Le nombre d'exécutions de tâches dont le statut est passé `CANCELED` au cours d'une période déterminé e par CloudWatch. (Pour plus d'informations sur CloudWatch les métriques, consultez Amazon CloudWatch Metrics.) La dimension `JobId` contient l'ID de la tâche.
`CanceledJobExecutionTotalCo unt`	Nombre total d'exécutions de tâche dont le statut est `CANCELED` pour la tâche donnée. La dimension `JobId` contient l'ID de la tâche.
`ClientErrorCount`	Nombre d'erreurs client générées pendant l'exécution de la tâche. La dimension `JobId` contient l'ID de la tâche.
`FailedJobExecutionCount`	Le nombre d'exécutions de tâches dont le statut est passé `FAILED` au cours d'une période déterminé e par CloudWatch. (Pour plus d'informations sur CloudWatch les métriques, consultez Amazon CloudWatch Metrics.) La dimension `JobId` contient l'ID de la tâche.
`FailedJobExecutionTotalCount`	Nombre total d'exécutions de tâche dont le statut est `FAILED` pour la tâche donnée. La dimension `JobId` contient l'ID de la tâche.
`InProgressJobExecutionCount`	Le nombre d'exécutions de tâches dont le statut est passé `IN_PROGRESS` au cours d'une période déterminée par CloudWatch. (Pour plus d'informa tions sur CloudWatch les métriques, consultez Amazon CloudWatch Metrics.) La dimension `JobId` contient l'ID de la tâche.

Métrique	Description
`InProgressJobExecutionTotalCount`	Nombre total d'exécutions de tâche dont le statut est `IN_PROGRESS` pour la tâche donnée. La dimension `JobId` contient l'ID de la tâche.
`RejectedJobExecutionTotalCount`	Nombre total d'exécutions de tâche dont le statut est `REJECTED` pour la tâche donnée. La dimension `JobId` contient l'ID de la tâche.
`RemovedJobExecutionTotalCount`	Nombre total d'exécutions de tâche dont le statut est `REMOVED` pour la tâche donnée. La dimension `JobId` contient l'ID de la tâche.
`QueuedJobExecutionCount`	Le nombre d'exécutions de tâches dont le statut est passé `QUEUED` au cours d'une période déterminée par CloudWatch. (Pour plus d'informations sur CloudWatch les métriques, consultez Amazon CloudWatch Metrics.) La dimension `JobId` contient l'ID de la tâche.
`QueuedJobExecutionTotalCount`	Nombre total d'exécutions de tâche dont le statut est `QUEUED` pour la tâche donnée. La dimension `JobId` contient l'ID de la tâche.
`RejectedJobExecutionCount`	Le nombre d'exécutions de tâches dont le statut est passé `REJECTED` au cours d'une période déterminée par CloudWatch. (Pour plus d'informations sur CloudWatch les métriques, consultez Amazon CloudWatch Metrics.) La dimension `JobId` contient l'ID de la tâche.
`RemovedJobExecutionCount`	Le nombre d'exécutions de tâches dont le statut est passé `REMOVED` au cours d'une période déterminée par CloudWatch. (Pour plus d'informations sur CloudWatch les métriques, consultez Amazon CloudWatch Metrics.) La dimension `JobId` contient l'ID de la tâche.

Métrique	Description
`ServerErrorCount`	Nombre d'erreurs de serveur générées pendant l'exécution de la tâche. La dimension `JobId` contient l'ID de la tâche.
`SuccededJobExecutionCount`	Le nombre d'exécutions de tâches dont le statut est passé `SUCCESS` au cours d'une période déterminé e par CloudWatch. (Pour plus d'informations sur CloudWatch les métriques, consultez Amazon CloudWatch Metrics.) La dimension `JobId` contient l'ID de la tâche.
`SuccededJobExecutionTotalCo unt`	Nombre total d'exécutions de tâche dont le statut est `SUCCESS` pour la tâche donnée. La dimension `JobId` contient l'ID de la tâche.

Métriques d'audit Device Defender

Métrique	Description
`NonCompliantResources`	Nombre de ressources détectées comme non conformes avec un contrôle. Le système renvoie le nombre de ressources non conformes pour chaque contrôle de chaque audit effectué.
`ResourcesEvaluated`	Nombre de ressources évaluées pour conformité. Le système renvoie le nombre de ressources évaluées pour chaque contrôle de chaque audit effectué.
`MisconfiguredDeviceDefender Notification`	Vous avertit lorsque votre configuration SNS pour AWS IoT Device Defender est mal configurée. Dimensions

Métriques de détection Device Defender

Métrique	Description
`NumOfMetricsExported`	Le nombre de métriques exportées pour une métrique côté cloud, côté appareil ou personnalisée. Le système indique le nombre de mesures exportées pour le compte, pour une métrique spécifique. Cette métrique est disponible uniquement pour les clients utilisant l'exportation de métriques.
`NumOfMetricsSkipped`	Le nombre de métriques ignorées pour une métrique côté cloud, côté appareil ou personnalisée. Le système indique le nombre de métriques ignorées pour le compte, pour une métrique spécifique en raison d'autorisations insuffisantes accordées au Device Defender Detect pour publier dans la rubrique mqtt. Cette métrique est disponible uniquement pour les clients utilisant l'exportation de métriques.
`NumOfMetricsExceedingSizeLimit`	Nombre de métriques ignorées lors de l'exportation pour une métrique côté cloud, côté appareil ou personnalisée en raison d'une taille dépassant les contraintes de taille des messages MQTT. Le système indique le nombre de métriques ignorées lors de l'exportation pour le compte, pour une métrique spécifique en raison d'une taille dépassant les contraintes de taille des messages MQTT. Cette métrique est disponible uniquement pour les clients utilisant l'exportation de métriques.
`Violations`	Nombre de nouvelles violations de comportement de profil de sécurité détectées depuis la dernière évaluation. Le système signale le nombre de nouvelles violations pour le compte, pour un profil de sécurité spécifique et pour un comportement spécifique d'un profil de sécurité spécifique.

Métrique	Description
`ViolationsCleared`	Nombre de violations de comportements de profil de sécurité résolues depuis la dernière évaluation. Le système signale le nombre de violations résolues pour le compte, pour un profil de sécurité spécifiqu e et pour un comportement spécifique d'un profil de sécurité spécifique.
`ViolationsInvalidated`	Nombre de violations de comportement de profil de sécurité pour lesquelles les informations ne sont plus disponibles depuis la dernière évaluatio n (l'appareil de génération de rapports ayant arrêté de créer des rapports ou n'étant plus surveillé pour une raison quelconque). Le système signale le nombre de violations non validées pour la totalité du compte, pour un profil de sécurité spécifique et pour un comportement spécifique d'un profil de sécurité spécifique.
`MisconfiguredDeviceDefender Notification`	Vous avertit lorsque votre configuration SNS pour AWS IoT Device Defender est mal configurée. Dimensions

Métriques de mise en service d'appareils

AWS IoT Métriques relatives à l'approvisionnement de la flotte

Métrique	Description
`ApproximateNumberOfThingsRe gistered`	Le nombre d'objets enregistrés par le Fleet Provision ing. Bien que le décompte soit généralement précis, l'architecture distribuée de AWS IoT Core rend difficile le maintien d'un décompte précis des objets enregistrés.

Métrique	Description
	La statistique à utiliser pour cette métrique . • Maximum pour indiquer le nombre total d'objets enregistrés. Pour connaître le nombre d'éléments enregistrés pendant la fenêtre CloudWatch d'agrégation, consultez la `RegisterThingFailed` métrique. Dimensions : ClaimCertificateId
`CreateKeysAndCertificateFailed`	Nombre d'échecs survenus lors de l'appel de `CreateKeysAndCertificate` l'API MQTT. La métrique est émise dans les cas de réussite (valeur = 0) et d'échec (valeur = 1). Cette métrique peut être utilisée pour suivre le nombre de certificats créés et enregistrés pendant les fenêtres d'agrégation CloudWatch prises en charge, par exemple 5 minutes ou 1 heure. Les statistiques disponibles pour cette métrique sont les suivantes : • Somme pour indiquer le nombre d'appels échoués. • SampleCountpour indiquer le nombre total d'appels réussis et échoués.

Métrique	Description
`CreateCertificateFromCsrFailed`	Nombre d'échecs survenus lors de l'appel de `CreateCertificateFromCsr` l'API MQTT. La métrique est émise dans les cas de réussite (valeur = 0) et d'échec (valeur = 1). Cette métrique peut être utilisée pour suivre le nombre d'éléments enregistrés pendant les fenêtres d'agrégation CloudWatch prises en charge, par exemple 5 minutes ou 1 heure. Les statistiques disponibles pour cette métrique sont les suivantes : • Somme pour indiquer le nombre d'appels échoués. • SampleCountpour indiquer le nombre total d'appels réussis et échoués.

Métrique	Description
`RegisterThingFailed`	Nombre d'échecs survenus lors de l'appel de `RegisterThing` l'API MQTT. La métrique est émise dans les cas de réussite (valeur = 0) et d'échec (valeur = 1). Cette métrique peut être utilisée pour suivre le nombre d'élément s enregistrés pendant les fenêtres d'agrégation CloudWatch prises en charge, par exemple 5 minutes ou 1 heure. Pour le nombre total d'objets enregistrés, consultez la `ApproximateNumber0 fThingsRegistered` métrique. Les statistiques disponibles pour cette métrique sont les suivantes : • Somme pour indiquer le nombre d'appels échoués. • SampleCountpour indiquer le nombre total d'appels réussis et échoués. Dimensions : TemplateName

Just-in-time métriques de provisionnement

Métrique	Description
`ProvisionThing.ClientError`	Le nombre de fois qu'un périphérique n'a pas pu être approvisionné en raison d'une erreur du client. Par exemple, la politique spécifiée dans le modèle n'existait pas.
`ProvisionThing.ServerError`	Le nombre de fois qu'un appareil n'a pas pu être approvisionné en raison d'une erreur du serveur. Les clients peuvent réessayer d'approvisionner l'appareil après avoir patienté et ils peuvent contacter AWS IoT si le problème persiste.

Métrique	Description
`ProvisionThing.Success`	Nombre de fois qu'un appareil a été approvisionné avec succès.

LoRaMétriques WAN

Le tableau suivant présente les mesures pour AWS IoT Core for LoRaWAN. Pour plus d'informations, consultez la section AWS IoT Core for LoRaWAN Mesures.

AWS IoT Core for LoRaWAN métriques

Métrique	Description
Appareils/passerelles actifs	Le nombre d'appareils LoRa WAN et de passerelles actifs dans votre compte.
Nombre de messages Uplink	Le nombre de messages de liaison montante envoyés pendant une durée spécifiée pour toutes les passerelles et tous les appareils actifs de votre. Compte AWS Les messages Uplink sont des messages envoyés depuis votre appareil à AWS IoT Core for LoRaWAN.
Nombre de messages en lien descendant	Le nombre de messages de liaison descendante envoyés pendant une durée spécifiée pour toutes les passerelles et appareils actifs de votre. Compte AWS Les messages de liaison descendante sont des messages AWS IoT Core for LoRaWAN envoyés depuis votre appareil.
Taux de perte de messages	Une fois que vous avez ajouté votre appareil et que vous vous y êtes connecté AWS IoT Core for LoRaWAN, celui-ci peut lancer un message en liaison montante pour commencer à échanger des messages avec le cloud. Vous pouvez utiliser cette métrique pour suivre ensuite le taux de perte de messages en liaison montante.

Métrique	Description
Joindre les métriques	Après avoir ajouté votre appareil et votre passerelle, vous devez exécuter une procédure de connexion afin que votre appareil puisse envoyer des données de liaison montante et communiquer avec AWS IoT Core for LoRaWAN. Vous pouvez utiliser cette métrique pour obtenir des informations sur les métriques d'adhésion pour tous les appareils actifs de votre Compte AWS.
Indicateur d'intensité moyenne du signal reçu (RSSI)	Vous pouvez utiliser cette métrique pour surveiller le RSSI moyen (indicateur d'intensité du signal reçu) pendant la durée spécifiée. Le RSSI est une mesure qui indique si le signal est suffisamment fort pour une bonne connexion sans fil. Cette valeur est négative et doit être proche de zéro pour une connexion solide.
Rapport signal/bruit moyen (SNR)	Vous pouvez utiliser cette métrique pour surveiller le rapport signal/bruit moyen (Signal-to-noise ratio) pendant la durée spécifiée. Le SNR est une mesure qui indique si le signal reçu est suffisamment fort par rapport au niveau de bruit pour une bonne connexion sans fil. La valeur SNR est positive et doit être supérieure à zéro pour indiquer que la puissance du signal est supérieure à la puissance du bruit.
Disponibilité de la passerelle	Vous pouvez utiliser cette métrique pour obtenir des informations sur la disponibilité de cette passerelle dans un délai spécifié. Cette métrique affiche le temps de connexion au websocket de cette passerelle pendant une durée spécifiée.

Just-in-time métriques de provisionnement

Métrique	Description
`ProvisionThing.ClientError`	Le nombre de fois qu'un périphérique n'a pas pu être approvisionné en raison d'une erreur du client. Par exemple, la politique spécifiée dans le modèle n'existait pas.
`ProvisionThing.ServerError`	Le nombre de fois qu'un appareil n'a pas pu être approvisionné en raison d'une erreur du serveur. Les clients peuvent réessayer d'approvisionner l'appareil après avoir patienté et ils peuvent contacter AWS IoT si le problème persiste.
`ProvisionThing.Success`	Nombre de fois qu’un appareil a été approvisionné avec succès.

Métriques d'indexation de la flotte

AWS IoT métriques d'indexation de la flotte

Métrique	Description
`NamedShadowCountForDynamicGroupQueryLimitExceeded`	Un maximum de 25 ombres nommées par objet sont traitées pour les termes de requête qui ne sont pas spécifiques à une source de données dans les groupes d'objets dynamiques. Lorsque cette limite est dépassée pour un objet, le `NamedShadowCountForDynamicGroupQueryLimitExceeded` type d'événement est émis.

Dimensions pour les métriques

Les métriques utilisent l'espace de noms et fournissent des métriques pour les dimensions suivantes.

Dimension	Description
`ActionType`	Le type d'action spécifié par la règle déclenchée par la demande.
`BehaviorName`	Nom du comportement du profil de sécurité de détection Device Defender qui est surveillé.
`ClaimCertificateId`	La `certificateId` de la réclamation utilisée pour approvisionner les appareils.
`CheckName`	Nom du contrôle d'audit Device Defender dont les résultats sont surveillés.
`JobId`	ID de la tâche dont la progression ou la connexion aux messages success/failure est surveillée.
`Protocol`	Protocole utilisé pour effectuer la demande. Les valeurs valides sont : MQTT ou HTTP
`RuleName`	Nom de la règle déclenchée par la demande.
`ScheduledAuditName`	Nom de l'audit Device Defender programmé dont les résultats du contrôle sont surveillés. La valeur `OnDemand` est utilisée si les résultats concernent un audit effectué à la demande.
`SecurityProfileName`	Nom du profil de sécurité de détection Device Defender dont les comportements sont surveillés.
`TemplateName`	Nom du modèle de mise en service.
`SourceArn`	Fait référence au profil de sécurité pour la détection ou à l'ARN du compte pour l'audit.

Dimension	Description
`RoleArn`	Fait référence au rôle que Device Defender a tenté d'assumer.
`TopicArn`	Fait référence à la rubrique SNS sur laquelle Device Defender a tenté de publier.
`Error`	Fournit une brève description de l'erreur reçue lors de la tentative de publication sur la rubrique SNS. Les valeurs possibles sont : • KMSKeyNotFound« » : indique que la clé KMS n'existe pas pour le sujet. • InvalidTopicName« » : indique que la rubrique SNS n'est pas valide. • « KMSAccess Refusé » : indique que le rôle n'est pas autorisé à accéder à la clé KMS du sujet. • AuthorizationError« » : indique que le rôle fourni n'autorise pas Device Defender à publier sur le sujet SNS. • SNSTopicNotFound« » : indique que la rubrique SNS fournie n'existe pas. • FailureToAssumeRole« » : indique que le rôle fourni n'autorise pas Device Defender à assumer le rôle. • CrossRegionSNSTopic« » : indique que le sujet SNS existe dans une autre région.

Surveiller AWS IoT à l'aide CloudWatch des journaux

Lorsque la AWS IoT journalisation est activée, AWS IoT envoie des événements de progression concernant chaque message lors de son transfert depuis vos appareils via le courtier de messages et le moteur de règles. Dans la CloudWatch console, CloudWatch les journaux apparaissent dans un groupe de AWSIotjournaux nommé Logs.

Pour plus d'informations sur les CloudWatch journaux, consultez la section CloudWatch Journaux. Pour plus d'informations sur AWS IoT CloudWatch les journaux pris en charge, consultezCloudWatch Enregistre les entrées du AWS IoT journal.

Afficher AWS IoT les journaux dans la CloudWatch console

> **Note**
>
> Le groupe de `AWSIotLogsV2` journaux n'est pas visible dans la CloudWatch console jusqu'à ce que :
>
> - Vous avez activé la connexion AWS IoT. Pour plus d'informations sur la façon d'activer la connexion AWS IoT, voir Configuration de la AWS IoT journalisation
> - Certaines entrées du journal ont été écrites par AWS IoT les opérations.

Pour consulter vos AWS IoT journaux dans la CloudWatch console

1. naviguez jusqu'à https://console.aws.amazon.com/cloudwatch/. Dans le panneau de navigation, choisissez Groupes de journaux.
2. Dans la zone de texte Filtre, entrez **`AWSIotLogsV2`**, puis appuyez sur Entrée.
3. Double-cliquez sur le `AWSIotLogsV2` groupe de journaux.
4. Choisissez Rechercher tout. La liste complète des AWS IoT journaux générés pour votre compte s'affiche.
5. Choisissez l'icône de développement pour afficher un flux individuel.

Vous pouvez également entrer une requête dans la zone de texte Filtrer les événements. Voici quelques demandes intéressantes à essayer :

- `{ $.logLevel = "INFO" }`

  Trouvez tous les journaux qui ont un niveau de journalisation `INFO`.

- `{ $.status = "Success" }`

  Trouvez tous les journaux qui ont un statut `Success`.

- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

Trouvez tous les journaux qui ont un statut `Success` et un type d'événement `GetThingShadow`.

Pour plus d'informations sur la création d'expressions de filtre, consultez la section CloudWatch Logs Requêtes.

CloudWatch Enregistre les entrées du AWS IoT journal

Chaque composant de AWS IoT génère ses propres entrées de journal. Chaque entrée de journal a un `eventType` qui spécifie l'opération ayant provoqué la génération de l'entrée de journal. Cette section décrit les entrées de journal générées par les AWS IoT composants suivants :

Rubriques

- Entrées du journal du courtier de messages
- Entrées du journal OCSP du certificat de serveur
- Entrées de journal de shadow d'appareil
- Entrées de journal du moteur de règles
- Entrées du journal des tâches
- Entrées de journal de provisionnement des appareils
- Entrées dynamiques du journal des groupes d'objets
- Entrées du journal d'indexation de la flotte
- Attributs communs CloudWatch des journaux

Entrées du journal du courtier de messages

Le courtier de AWS IoT messages génère des entrées de journal pour les événements suivants :

Rubriques

- Connexion de l'entrée de journal
- Déconnecter l'entrée du journal
- GetRetainedMessage entrée dans le journal
- ListRetainedMessage entrée dans le journal
- Entrée de journal Publish-In
- Entrée du journal Publish-Out

- Entrée de journal en file d'attente
- Souscrire l'entrée de journal
- Entrée dans le journal de désabonnement

Connexion de l'entrée de journal

Le courtier de AWS IoT messages génère une entrée de journal avec un « `eventType` of » `Connect` lorsqu'un client MQTT se connecte.

Exemple d'entrée de journal de connexion

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Connect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `Connect` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

sourceIp

L'adresse IP d'origine de la demande.

sourcePort

Le port d'origine de la demande.

Déconnecter l'entrée du journal

Le courtier de AWS IoT messages génère une entrée de journal avec un « `eventType` of » `Disconnect` lorsqu'un client MQTT se déconnecte.

Exemple d'entrée de journal de déconnexion

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID",
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `Disconnect` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

sourceIp

L'adresse IP d'origine de la demande.

sourcePort

Le port d'origine de la demande.

raison

La raison pour laquelle le client se déconnecte.

détails

Une brève explication de l'erreur.

disconnectReason

La raison pour laquelle le client se déconnecte.

GetRetainedMessage entrée dans le journal

Le courtier de AWS IoT messages génère une entrée de journal avec un « `eventType` of `GetRetainedMessage` GetRetainedMessagewhen ».

GetRetainedMessage exemple de saisie de journal

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetRetainedMessage",
    "protocol": "HTTP",
    "topicName": "a/b/c",
    "qos": "1",
    "lastModifiedDate": "2017-08-07 18:47:56.664"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `GetRetainedMessage` contiennent les attributs suivants :

lastModifiedDate

Date et heure de l'époque, en millisecondes, auxquelles le message conservé a été stocké par. AWS IoT

protocole ;

Protocole utilisé pour effectuer la demande. Valeur valide : `HTTP`.

qos

Le niveau de Qualité de Service (QoS) utilisé dans la demande de publication. Les valeurs valides sont `0` ou `1`.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

ListRetainedMessage entrée dans le journal

Le courtier de AWS IoT messages génère une entrée de journal avec un « `eventType` of `ListRetainedMessage` `ListRetainedMessages`when ».

ListRetainedMessage exemple de saisie de journal

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ListRetainedMessage",
    "protocol": "HTTP"
}
```

En plus de Attributs communs CloudWatch des journaux, `ListRetainedMessage` les entrées de journal contiennent les attributs suivants :

protocole ;

Protocole utilisé pour effectuer la demande. Valeur valide : `HTTP`.

Entrée de journal Publish-In

Lorsque le courtier de AWS IoT messages reçoit un message MQTT, il génère une entrée de journal avec un `eventType` de`Publish-In`.

Exemple d'entrée de journal Publish-In

```
{
        "timestamp": "2017-08-10 15:39:30.961",
        "logLevel": "INFO",
        "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
        "accountId": "123456789012",
        "status": "Success",
        "eventType": "Publish-In",
        "protocol": "MQTT",
        "topicName": "$aws/things/MyThing/shadow/get",
        "clientId": "abf27092886e49a8a5c1922749736453",
        "principalId":
 "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
        "sourceIp": "205.251.233.181",
        "sourcePort": 13490,
        "retain": "True"
    }
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `Publish-In` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

retain

Attribut utilisé lorsque l'indicateur RETAIN est défini sur un message avec la valeur de `True`. Si l'indicateur RETAIN n'est pas activé sur le message, cet attribut n'apparaît pas dans l'entrée du journal. Pour de plus amples informations, veuillez consulter MQTT a retenu les messages.

sourceIp

L'adresse IP d'origine de la demande.

sourcePort

Le port d'origine de la demande.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

Entrée du journal Publish-Out

Lorsque le courtier de messages publie un message MQTT, il génère une entrée de journal avec un `eventType` de `Publish-Out`

Exemple d'entrée de journal Publish-Out

```
{
    "timestamp": "2017-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Publish-Out",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/get",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `Publish-Out` contiennent les attributs suivants :

clientId

L'ID du client abonné qui reçoit des messages sur ce sujet MQTT.

principalId

L'ID du mandataire formulant la demande.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

sourceIp

L'adresse IP d'origine de la demande.

sourcePort

Le port d'origine de la demande.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

Entrée de journal en file d'attente

Lorsqu'un appareil doté d'une session persistante est déconnecté, le courtier de messages MQTT stocke les messages du périphérique et AWS IoT génère des entrées de journal avec un `Queued` EventType de. Pour de plus amples informations sur les session persistantes MQTT, veuillez consulter Sessions permanentes MQTT.

Exemple d'entrée dans le journal des erreurs d'un serveur en file d'attente

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Server Error"
}
```

En plus de Attributs communs CloudWatch des journaux, `Queued` les erreurs de serveur des entrées de journal contiennent les attributs suivants :

clientId

ID du client auquel le message est mis en file d'attente.

détails

`Server Error`

Une erreur du serveur a empêché le stockage du message.

protocole ;

Protocole utilisé pour effectuer la demande. Cette valeur sera toujours `MQTT`.

qos

Niveau de Qualité de Service (QoS) de la requête. La valeur sera toujours 1 car les messages dont le QoS est égal à 0 ne sont pas stockés.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

Exemple d'entrée dans le journal des succès en attente

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Success"
}
```

En plus de Attributs communs CloudWatch des journaux, `Queued` les entrées de journal contiennent les attributs suivants :

clientId

ID du client auquel le message est mis en file d'attente.

protocole ;

Protocole utilisé pour effectuer la demande. Cette valeur sera toujours `MQTT`.

qos

Niveau de Qualité de Service (QoS) de la requête. La valeur sera toujours 1 car les messages dont le QoS est égal à 0 ne sont pas stockés.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

Exemple d'entrée de journal limitée en file d'attente

```
{
    "timestamp": "2022-08-10 15:39:30.961",
    "logLevel": "ERROR",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "topicName": "$aws/things/MyThing/get",
    "clientId": "123123123",
    "qos": "1",
    "protocol": "MQTT",
    "eventType": "Queued",
    "status": "Failure",
    "details": "Throttled while queueing offline message"
}
```

En plus de Attributs communs CloudWatch des journaux, `Queued` les entrées de journal limitées contiennent les attributs suivants :

clientId

ID du client auquel le message est mis en file d'attente.

détails

`Throttled while queueing offline message`

Le client a dépassé la `Queued messages per second per account` limite, le message n'a donc pas été enregistré.

protocole ;

Protocole utilisé pour effectuer la demande. Cette valeur sera toujours `MQTT`.

qos

Niveau de Qualité de Service (QoS) de la requête. La valeur sera toujours 1 car les messages dont le QoS est égal à 0 ne sont pas stockés.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

Souscrire l'entrée de journal

Le courtier de AWS IoT messages génère une entrée de journal avec un « `eventType` of » `Subscribe` lorsqu'un client MQTT s'abonne à un sujet.

Exemple d'entrée de journal d'abonnement MQTT 3

```
{
    "timestamp": "2017-08-10 15:39:04.413",
    "logLevel": "INFO",
    "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Subscribe",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/#",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `Subscribe` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

protocole ;

Protocole utilisé pour effectuer la demande. Cette valeur sera toujours `MQTT`.

sourceIp

L'adresse IP d'origine de la demande.

sourcePort

Le port d'origine de la demande.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

Exemple d'entrée de journal d'abonnement MQTT 5

```
{
 "timestamp": "2022-11-30 16:24:15.628",
 "logLevel": "INFO",
 "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "Subscribe",
 "protocol": "MQTT",
 "topicName": "test/topic1,$invalid/reserved/topic",
 "subscriptions": [
  {
   "topicName": "test/topic1",
   "reasonCode": 1
  },
  {
   "topicName": "$invalid/reserved/topic",
   "reasonCode": 143
  }
 ],
 "clientId": "abf27092886e49a8a5c1922749736453",
 "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
 "sourceIp": "205.251.233.181",
 "sourcePort": 13490
}
```

Pour les opérations d'abonnement de MQTT 5, en plus desAttributs communs CloudWatch des journaux et des attributs d'entrée du journal d'abonnement et de MQTT 3, les entrées du `Subscribe` journal MQTT 5 contiennent l'attribut suivant :

abonnements

Une liste de correspondances entre les sujets demandés dans la demande d'abonnement et le code de motif individuel de MQTT 5. Pour plus d'informations, consultez code de motif MQTT.

Entrée dans le journal de désabonnement

Le courtier de AWS IoT messages génère une entrée de journal avec un « `eventType` of » `Unsubscribe` lorsqu'un client MQTT se désabonne d'un sujet MQTT.

Exemple d'entrée dans le journal de désabonnement MQTT

```
{
    "timestamp": "2024-08-20 22:53:32.844",
    "logLevel": "INFO",
    "traceId": "db6bd09a-2c3f-1cd2-27cc-fd6b1ce03b58",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Unsubscribe",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `Unsubscribe` contiennent les attributs suivants :

protocole ;

Protocole utilisé pour effectuer la demande. Cette valeur sera toujours `MQTT`.

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

sourceIp

L'adresse IP d'origine de la demande.

sourcePort

Le port d'origine de la demande.

Entrées du journal OCSP du certificat de serveur

AWS IoT Core génère des entrées de journal pour l'événement suivant :

Rubriques

- Récupérer l'entrée du journal de OCSPStaple données
- Récupérer l'entrée du journal de OCSPStaple données pour les points de terminaison privés

Récupérer l'entrée du journal de OCSPStaple données

AWS IoT Core génère une entrée de journal avec un `eventType` de `RetrieveOCSPStapleData` lorsque le serveur récupère les données de base OCSP.

Exemples d'entrées dans le journal de récupération des OCSPStaple données

Voici un exemple d'entrée de journal de`Success`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "200",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
```

```
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
  "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```

Voici un exemple d'entrée de journal de`Failure`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "A non 2xx HTTP response was received from the OCSP responder.",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
 "connectionDetails": {
  "httpStatusCode": "444",
  "ocspResponderUri": "http://ocsp.example.com",
  "sourceIp": "205.251.233.181",
  "targetIp": "250.15.5.3"
 },
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```

Pour l'`RetrieveOCSPStaple`opération, outre leAttributs communs CloudWatch des journaux, les entrées du journal contiennent les attributs suivants :

raison

La raison pour laquelle l'opération échoue.

domainConfigName

Le nom de la configuration de votre domaine.

Détails de connexion

Brève explication des détails de connexion.

- httpStatusCode

  Codes d'état HTTP renvoyés par le répondeur OCSP en réponse à la demande du client adressée au serveur.

- ocspResponderUri

  L'URI du répondeur OCSP qui AWS IoT Core extrait le certificat du serveur.

- sourceIp

  Adresse IP source du AWS IoT Core serveur.

- IP cible

  Adresse IP cible du répondeur OCSP.

ocspRequestDetails

Détails de la demande OCSP.

- Nom du demandeur

  Identifiant du AWS IoT Core serveur qui envoie une demande au répondeur OCSP.

- requestCertId

  L'ID du certificat de la demande. Il s'agit de l'ID du certificat pour lequel la réponse OCSP est demandée.

ocspResponseDetails

Détails de la réponse de l'OCSP.

- responseCertId

ID de certificat de la réponse OCSP.

- ocspResponseStatus

  État de la réponse OCSP.

- État du certificat

  État du certificat.

- Signature

  Signature appliquée à la réponse par une entité de confiance.

- thisUpdateTime

  L'heure à laquelle le statut indiqué est connu pour être correct.

- nextUpdateTime

  Heure à laquelle ou avant laquelle les informations les plus récentes concernant le statut du certificat seront disponibles.

- producedAtTime

  Heure à laquelle le répondeur OCSP a signé cette réponse.

- stapledDataPayloadTaille

  La taille de la charge utile des données agrafées.

Récupérer l'entrée du journal de OCSPStaple données pour les points de terminaison privés

AWS IoT Core génère une entrée de journal avec un `eventType` de `RetrieveOCSPStapleData` lorsque le serveur récupère les données de base OCSP.

Exemples de saisie OCSPStaple de journaux de données pour les points de terminaison privés

Voici un exemple d'entrée de journal de`Success`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "INFO",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Success",
```

```
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
 "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 },
 "ocspResponseDetails": {
  "responderId": "04:C1:3F:8F:27:D6:49:13:F8:DE:B2:36:9D:85:8E:F8:31:3B:A6:D0"
              "responseCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
  "ocspResponseStatus": "successful",
  "certStatus": "good",
  "signature":
 "4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  "thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
  "nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
  "producedAtTime": "Jan 31 01:37:03 2024 UTC",
  "stapledDataPayloadSize": "XXX"
 }
}
```

Voici un exemple d'entrée de journal de`Failure`.

```
{
 "timestamp": "2024-01-30 15:39:30.961",
 "logLevel": "ERROR",
 "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
 "accountId": "123456789012",
 "status": "Failure",
 "reason": "The payload returned by the Lambda function exceeds the maximum response
 size of 7 kilobytes.",
 "eventType": "RetrieveOCSPStapleData",
 "domainConfigName": "test-domain-config-name",
        "lambdaDetails": {
        "lambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:my-function",
```

```
        "sourceArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
testDomainConfigure/6bzfg"
      },
        "authorizedResponderArn": "arn:aws:acm:us-west-2:123456789012:certificate/
certificate_ID",
  "ocspRequestDetails": {
  "requesterName": "iot.amazonaws.com",
  "requestCertId":
 "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:
 }
}
```


Pour l'`RetrieveOCSPStaple`opération, outre les attributs Attributs communs CloudWatch des journaux et contenus dans l'entrée du journal Retrieve OCSPStaple Data, les entrées du journal des points de terminaison privés contiennent les attributs suivants :

Détails de la Lambda

Détails de la fonction Lambda.

- Fil d'agneau

  L'ARN de la fonction Lambda.

- Source ARN

  ARN de la configuration du domaine.

authorizedResponderArn

L'ARN du répondeur d'autorisation s'il en existe un configuré dans la configuration du domaine.

Entrées de journal de shadow d'appareil

Le service AWS IoT Device Shadow génère des entrées de journal pour les événements suivants :

Rubriques

- DeleteThingShadow entrée dans le journal
- GetThingShadow entrée dans le journal
- UpdateThingShadow entrée dans le journal

DeleteThingShadow entrée dans le journal

Le service Device Shadow génère une entrée de journal avec un `eventType` de `DeleteThingShadow` en cas de réception d'une demande de suppression d'un shadow d'appareil.

DeleteThingShadow exemple de saisie de journal

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/delete"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `DeleteThingShadow` contiennent les attributs suivants :

deviceShadowName

Nom du shadow à mettre à jour.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

topicName

Le nom de la rubrique sur laquelle la demande a été publiée.

GetThingShadow entrée dans le journal

Le service Device Shadow génère une entrée de journal avec un `eventType` de `GetThingShadow` en cas de réception d'une demande get pour un shadow.

GetThingShadow exemple de saisie de journal

```
{
    "timestamp": "2017-08-09 17:56:30.941",
```

```
    "logLevel": "INFO",
    "traceId": "b575f19a-97a2-cf72-0ed0-c64a783a2504",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "MyThing",
    "topicName": "$aws/things/MyThing/shadow/get"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `GetThingShadow` contiennent les attributs suivants :

deviceShadowName

Le nom du shadow demandé.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

topicName

Le nom de la rubrique sur laquelle la demande a été publiée.

UpdateThingShadow entrée dans le journal

Le service Device Shadow génère une entrée de journal avec un `eventType` de `UpdateThingShadow` en cas de réception d'une demande de mise à jour d'un shadow d'appareil.

UpdateThingShadow exemple de saisie de journal

```
{
    "timestamp": "2017-08-07 18:43:59.436",
    "logLevel": "INFO",
    "traceId": "d0074ba8-0c4b-a400-69df-76326d414c28",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/update"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `UpdateThingShadow` contiennent les attributs suivants :

deviceShadowName

Nom du shadow à mettre à jour.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

topicName

Le nom de la rubrique sur laquelle la demande a été publiée.

Entrées de journal du moteur de règles

Le moteur de AWS IoT règles génère des journaux pour les événements suivants :

Rubriques

- FunctionExecution entrée dans le journal
- RuleExecution entrée dans le journal
- RuleMatch entrée dans le journal
- RuleExecutionThrottled entrée dans le journal
- RuleNotFound entrée dans le journal
- StartingRuleExecution entrée dans le journal

FunctionExecution entrée dans le journal

Le moteur de règles génère une entrée de journal avec un `eventType` de `FunctionExecution` lorsque la requête SQL d'une règle appelle une fonction externe. Une fonction externe est appelée lorsque l'action d'une règle envoie une requête HTTP à AWS IoT un autre service Web (par exemple, en appelant `get_thing_shadow` ou`machinelearning_predict`).

FunctionExecution exemple de saisie de journal

```
{
    "timestamp": "2017-07-13 18:33:51.903",
    "logLevel": "DEBUG",
    "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
```

```
    "status": "Success",
    "eventType": "FunctionExecution",
    "clientId": "N/A",
    "topicName":"rules/test",
    "ruleName": "ruleTestPredict",
    "ruleAction": "MachinelearningPredict",
    "resources": {
        "ModelId": "predict-model"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `FunctionExecution` contiennent les attributs suivants :

clientId

N/A pour les journaux `FunctionExecution`.

principalId

L'ID du mandataire formulant la demande.

resources

Une collection des ressources utilisées par les actions de la règle.

ruleName

Le nom de la règle de correspondance.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

RuleExecution entrée dans le journal

Lorsque le moteur de AWS IoT règles déclenche l'action d'une règle, il génère une entrée de `RuleExecution` journal.

RuleExecution exemple de saisie de journal

```
{
    "timestamp": "2017-08-10 16:32:46.070",
    "logLevel": "INFO",
```

```
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "resources": {
        "RepublishTopic": "rules/republish"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```


En plus de Attributs communs CloudWatch des journaux, les entrées de journal `RuleExecution` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

resources

Une collection des ressources utilisées par les actions de la règle.

ruleAction

Le nom de l'action déclenchée.

ruleName

Le nom de la règle de correspondance.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

RuleMatch entrée dans le journal

Le moteur de AWS IoT règles génère une entrée de journal avec un `eventType` de `RuleMatch` lorsque le courtier de messages reçoit un message correspondant à une règle.

RuleMatch exemple de saisie de journal

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleMatch",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `RuleMatch` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

ruleName

Le nom de la règle de correspondance.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

RuleExecutionThrottled entrée dans le journal

Lorsqu'une exécution est limitée, le moteur de AWS IoT règles génère une entrée de journal avec un `eventType` de. `RuleExecutionThrottled`

RuleExecutionThrottled exemple de saisie de journal

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
```

```
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecutionThrottled",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleExecutionThrottled",
    "details": "Exection of Rule example_rule throttled"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `RuleExecutionThrottled` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

détails

Une brève explication de l'erreur.

principalId

L'ID du mandataire formulant la demande.

raison

La chaîne « RuleExecutionThrottled ».

ruleName

Le nom de la règle à déclencher.

topicName

Le nom de la rubrique qui a été publiée.

RuleNotFound entrée dans le journal

Lorsque le moteur de AWS IoT règles ne trouve pas de règle portant un nom donné, il génère une entrée de journal avec un « `eventType` of `RuleNotFound` ».

RuleNotFound exemple de saisie de journal

```
{
```

```
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleNotFound",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleNotFound",
    "details": "Rule example_rule not found"
}
```


En plus de Attributs communs CloudWatch des journaux, les entrées de journal `RuleNotFound` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

détails

Une brève explication de l'erreur.

principalId

L'ID du mandataire formulant la demande.

raison

La chaîne « RuleNotFound ».

ruleName

Nom de la règle qui est introuvable.

topicName

Le nom de la rubrique qui a été publiée.

StartingRuleExecution entrée dans le journal

Lorsque le moteur de AWS IoT règles commence à déclencher l'action d'une règle, il génère une entrée de journal avec un `eventType` de`StartingRuleExecution`.

StartingRuleExecution exemple de saisie de journal

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "DEBUG",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartingRuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `rule-` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

principalId

L'ID du mandataire formulant la demande.

ruleAction

Le nom de l'action déclenchée.

ruleName

Le nom de la règle de correspondance.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

Entrées du journal des tâches

Le service AWS IoT Job génère des entrées de journal pour les événements suivants. Les entrées de journal sont générées lorsqu'une demande HTTP ou MQTT est reçue à partir de l'appareil.

Rubriques

- DescribeJobExecution entrée dans le journal
- GetPendingJobExecution entrée dans le journal
- ReportFinalJobExecutionCount entrée dans le journal
- StartNextPendingJobExecution entrée dans le journal
- UpdateJobExecution entrée dans le journal

DescribeJobExecution entrée dans le journal

Le service AWS IoT Jobs génère une entrée de journal avec un « `eventType` of » `DescribeJobExecution` lorsque le service reçoit une demande décrivant l'exécution d'une tâche.

DescribeJobExecution exemple de saisie de journal

```
{
    "timestamp": "2017-08-10 19:13:22.841",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DescribeJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/get",
    "clientToken": "myToken",
    "details": "The request status is SUCCESS."
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `GetJobExecution` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

clientToken

Identifiant unique, sensible à la casse, afin de garantir l'idempotence de la demande. Pour de plus amples informations, veuillez consulter la section Comment garantir l'idempotence.

détails

Informations supplémentaires issues du service Jobs.

jobId

L'ID de tâche pour l'exécution du travail.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

topicName

La rubrique utilisée pour effectuer la demande.

GetPendingJobExecution entrée dans le journal

Le service AWS IoT Jobs génère une entrée de journal avec un « `eventType` of » `GetPendingJobExecution` lorsque le service reçoit une demande d'exécution de tâche.

GetPendingJobExecution exemple de saisie de journal

```
{
    "timestamp": "2018-06-13 17:45:17.197",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "299966ad-54de-40b4-99d3-4fc8b52da0c5",
    "topicName": "$aws/things/299966ad-54de-40b4-99d3-4fc8b52da0c5/jobs/get",
    "clientToken": "24b9a741-15a7-44fc-bd3c-1ff2e34e5e82",
    "details": "The request status is SUCCESS."
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `GetPendingJobExecution` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

clientToken

Identifiant unique, sensible à la casse, afin de garantir l'idempotence de la demande. Pour de plus amples informations, veuillez consulter la section Comment garantir l'idempotence.

détails

Informations supplémentaires issues du service Jobs.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

topicName

Le nom de la rubrique faisant l'objet de l'abonnement.

ReportFinalJobExecutionCount entrée dans le journal

Le service AWS IoT Jobs génère une entrée de journal avec un « `entryType` of » `ReportFinalJobExecutionCount` lorsqu'une tâche est terminée.

ReportFinalJobExecutionCount exemple de saisie de journal

```
{
    "timestamp": "2017-08-10 19:44:16.776",
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ReportFinalJobExecutionCount",
    "jobId": "002",
    "details": "Job 002 completed. QUEUED job execution count: 0 IN_PROGRESS job
 execution count: 0 FAILED job execution count: 0 SUCCEEDED job execution count: 1
 CANCELED job execution count: 0 REJECTED job execution count: 0 REMOVED job execution
 count: 0"
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `ReportFinalJobExecutionCount` contiennent les attributs suivants :

détails

Informations supplémentaires issues du service Jobs.

jobId

L'ID de tâche pour l'exécution du travail.

StartNextPendingJobExecution entrée dans le journal

Lorsqu'il reçoit une demande pour démarrer la prochaine exécution de la tâche en attente, le service AWS IoT Jobs génère une entrée de journal avec un `eventType` de`StartNextPendingJobExecution`.

StartNextPendingJobExecution exemple de saisie de journal

```
{
    "timestamp": "2018-06-13 17:49:51.036",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartNextPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "95c47808-b1ca-4794-bc68-a588d6d9216c",
    "topicName": "$aws/things/95c47808-b1ca-4794-bc68-a588d6d9216c/jobs/start-next",
    "clientToken": "bd7447c4-3a05-49f4-8517-dd89b2c68d94",
    "details": "The request status is SUCCESS."
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `StartNextPendingJobExecution` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

clientToken

Identifiant unique, sensible à la casse, afin de garantir l'idempotence de la demande. Pour de plus amples informations, veuillez consulter la section Comment garantir l'idempotence.

détails

Informations supplémentaires issues du service Jobs.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

topicName

La rubrique utilisée pour effectuer la demande.

UpdateJobExecution entrée dans le journal

Le service AWS IoT Jobs génère une entrée de journal avec un « `eventType` of » `UpdateJobExecution` lorsque le service reçoit une demande de mise à jour de l'exécution d'une tâche.

UpdateJobExecution exemple de saisie de journal

```
{
    "timestamp": "2017-08-10 19:25:14.758",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/update",
    "clientToken": "myClientToken",
    "versionNumber": "1",
    "details": "The destination status is IN_PROGRESS. The request status is SUCCESS."
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `UpdateJobExecution` contiennent les attributs suivants :

clientId

L'ID du client formulant la demande.

clientToken

Identifiant unique, sensible à la casse, afin de garantir l'idempotence de la demande. Pour de plus amples informations, veuillez consulter la section Comment garantir l'idempotence.

détails

Informations supplémentaires issues du service Jobs.

jobId

L'ID de tâche pour l'exécution du travail.

protocole ;

Protocole utilisé pour effectuer la demande. Les valeurs valides sont `MQTT` ou `HTTP`.

topicName

La rubrique utilisée pour effectuer la demande.

versionNumber

Version de l'exécution de tâche.

Entrées de journal de provisionnement des appareils

Le service AWS IoT Device Provisioning génère des journaux pour les événements suivants.

Rubriques

- GetDeviceCredentials entrée dans le journal
- ProvisionDevice entrée dans le journal

GetDeviceCredentials entrée dans le journal

Le service AWS IoT Device Provisioning génère une entrée de journal avec un « `eventType` of » `GetDeviceCredential` lorsqu'un client appelle`GetDeviceCredential`.

GetDeviceCredentialsexemple de saisie de journal

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "GetDeviceCredentials",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `GetDeviceCredentials` contiennent les attributs suivants :

détails

Une brève explication de l'erreur.

deviceCertificateId

ID du certificat d'appareil.

ProvisionDevice entrée dans le journal

Le service AWS IoT Device Provisioning génère une entrée de journal avec un « `eventType` of » `ProvisionDevice` lorsqu'un client appelle`ProvisionDevice`.

ProvisionDevice exemple de saisie de journal

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "ProvisionDevice",
  "provisioningTemplateName" : "myTemplate",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
 }
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `ProvisionDevice` contiennent les attributs suivants :

détails

Une brève explication de l'erreur.

deviceCertificateId

ID du certificat d'appareil.

provisioningTemplateName

Nom du modèle de mise en service.

Entrées dynamiques du journal des groupes d'objets

AWS IoT Les groupes d'objets dynamiques génèrent des journaux pour l'événement suivant.

Rubriques

- AddThingToDynamicThingGroupsFailed entrée dans le journal

AddThingToDynamicThingGroupsFailed entrée dans le journal

Lorsque vous ne AWS IoT parvenez pas à ajouter un élément aux groupes dynamiques spécifiés, il génère une entrée de journal avec un `eventType` de`AddThingToDynamicThingGroupsFailed`. Cela se produit lorsqu'un objet a répondu aux critères d'appartenance au groupe d'objets dynamiques. Toutefois, il n'a pas pu être ajouté au groupe dynamique ou il a été supprimé du groupe dynamique. Cela peut se produire pour les raisons suivantes :

- L'objet appartient déjà au nombre maximal de groupes.
- L'option --override-dynamic-groups a été utilisée pour ajouter l'objet à un groupe d'objets statiques. Il a été retiré d'un groupe d'objets dynamiques pour rendre cela possible.

Pour de plus amples informations, veuillez consulter Limitations et conflits de groupes d'objets dynamiques.

AddThingToDynamicThingGroupsFailed exemple de saisie de journal

Cet exemple montre une entrée de journal correspondant à une erreur `AddThingToDynamicThingGroupsFailed`. Dans cet exemple, il TestThingrépondait aux critères pour figurer dans les groupes d'objets dynamiques répertoriés dans`dynamicThingGroupNames`, mais n'a pas pu être ajouté à ces groupes dynamiques, comme décrit dans`reason`.

```
{
 "timestamp": "2020-03-16 22:24:43.804",
 "logLevel": "ERROR",
 "traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
 "accountId": "57EXAMPLE833",
 "status": "Failure",
 "eventType": "AddThingToDynamicThingGroupsFailed",
 "thingName": "TestThing",
 "dynamicThingGroupNames": [
  "DynamicThingGroup11",
  "DynamicThingGroup12",
```

```
  "DynamicThingGroup13",
  "DynamicThingGroup14"
 ],
 "reason": "The thing failed to be added to the given dynamic thing group(s) because
 the thing already belongs to the maximum allowed number of groups."
}
```

En plus de Attributs communs CloudWatch des journaux, les entrées de journal `AddThingToDynamicThingGroupsFailed` contiennent les attributs suivants :

dynamicThingGroupNoms

Tableau des groupes d'objets dynamiques auquel l'objet n'a pas pu être ajouté.

raison

Raison pour laquelle l'objet n'a pas pu être ajouté aux groupes d'objets dynamiques.

thingName

Nom de l'objet qui n'a pas pu être ajouté à un groupe d'objets dynamiques.

Entrées du journal d'indexation de la flotte

AWS IoT l'indexation de la flotte génère des entrées de journal pour les événements suivants.

Rubriques

- NamedShadowCountForDynamicGroupQueryLimitExceeded entrée dans le journal

NamedShadowCountForDynamicGroupQueryLimitExceeded entrée dans le journal

Un maximum de 25 ombres nommées par objet sont traitées pour les termes de requête qui ne sont pas spécifiques à une source de données dans les groupes dynamiques. Lorsque cette limite est dépassée pour un objet, le type d'`NamedShadowCountForDynamicGroupQueryLimitExceeded`événement est émis.

NamedShadowCountForDynamicGroupQueryLimitExceeded exemple de saisie de journal

Cet exemple montre une entrée de journal correspondant à une `NamedShadowCountForDynamicGroupQueryLimitExceeded` erreur. Dans cet exemple, tous les `DynamicGroup` résultats basés sur toutes les valeurs peuvent être inexacts, comme décrit dans le `reason` champ.

```
{
"timestamp": "2020-03-16 22:24:43.804",
"logLevel": "ERROR",
"traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
"accountId": "571032923833",
"status": "Failure",
"eventType": "NamedShadowCountForDynamicGroupQueryLimitExceeded",
"thingName": "TestThing",
"reason": "A maximum of 25 named shadows per thing are processed for non-data source
 specific query terms in dynamic groups."
}
```

Attributs communs CloudWatch des journaux

Toutes les entrées du journal CloudWatch Logs incluent les attributs suivants :

accountId

Votre Compte AWS identifiant.

eventType

Le type d'événement pour lequel le journal a été créé. La valeur du type d'événement dépend de l'événement qui a généré l'entrée de journal. Chaque description d'entrée de journal inclut la valeur de `eventType` pour cette entrée de journal.

logLevel

Le niveau de journalisation utilisé. Pour de plus amples informations, veuillez consulter the section called “Niveaux de journalisation.”.

status

Le statut d'une demande.

timestamp

L'horodatage UTC lisible par l'homme du moment où le client s'est connecté à AWS IoT l’agent de messages.

traceId

Un identifiant généré de façon aléatoire qui peut être utilisé pour mettre en corrélation tous les journaux pour une demande spécifique.

Téléchargez les journaux côté appareil sur Amazon CloudWatch

Vous pouvez télécharger l'historique des journaux côté appareil sur Amazon CloudWatch afin de surveiller et d'analyser l'activité d'un appareil sur le terrain. Les journaux côté appareil peuvent inclure des fichiers journaux du système, des applications et des appareils. Ce processus utilise un paramètre d'action CloudWatch Logs rules pour publier les journaux côté appareil dans un groupe de journaux défini par le client.

Comment ça marche

Le processus commence lorsqu'un AWS IoT appareil envoie des messages MQTT contenant des fichiers journaux formatés à un AWS IoT sujet. Une AWS IoT règle surveille le sujet du message et envoie les fichiers journaux à un groupe de CloudWatch journaux que vous définissez. Vous pouvez ensuite examiner et analyser les informations.

Rubriques

- Rubriques MQTT
- Action de la règle

Rubriques MQTT

Choisissez un espace de nom de rubrique MQTT que vous utiliserez pour publier les journaux. Nous vous recommandons d'utiliser ce format pour l'espace des rubriques communes, `$aws/rules/things/thing_name/logs`, et ce format pour les rubriques d'erreur,`$aws/rules/things/thing_name/logs/errors`. La structure de dénomination pour les journaux et les sujets d'erreur est recommandée, mais elle n'est pas obligatoire. Pour plus d'informations, consultez Designing MQTT Topics for AWS IoT Core.

En utilisant l'espace thématique commun recommandé, vous utilisez les rubriques réservées AWS IoT de Basic Ingest. AWS IoT Basic Ingest envoie en toute sécurité les données de l'appareil AWS aux services pris en charge par des actions de AWS IoT règles. Cela supprime le courtier de publish/subscribe messages du chemin d'ingestion, le rendant ainsi plus rentable. Pour de plus amples informations, veuillez consulter Reducing messaging costs with Basic Ingest.

Si vous utilisez batchMode pour télécharger des fichiers journaux, vos messages doivent suivre un format spécifique incluant un horodatage et un message UNIX. Pour plus d'informations, consultez la rubrique relative aux exigences relatives au format des messages MQTT pour le mode BatchMode dans la section Action des règles de CloudWatch journalisation.

Action de la règle

Lors de la AWS IoT réception des messages MQTT des appareils clients, une AWS IoT règle surveille le sujet défini par le client et publie le contenu dans un groupe de CloudWatch journaux que vous définissez. Ce processus utilise une action de règle CloudWatch Logs pour surveiller le MQTT pour détecter des lots de fichiers journaux. Pour plus d'informations, consultez l'action relative à la AWS IoT règle CloudWatch Logs.

Mode par lots

`batchMode`est un paramètre booléen inclus dans l'action de règle AWS IoT CloudWatch Logs. Ce paramètre est facultatif et est désactivé (`false`) par défaut. Pour télécharger des fichiers journaux côté appareil par lots, vous devez activer ce paramètre (`true`) lors de la création de la règle. AWS IoT Pour plus d'informations, consultez la section CloudWatch Journaux dans la section des actions relatives aux AWS IoT règles.

Téléchargement de journaux côté appareil à l'aide de AWS IoT règles

Vous pouvez utiliser le moteur de AWS IoT règles pour télécharger des enregistrements de journal à partir de fichiers journaux existants côté appareil (journaux du système, des applications et de l'appareil-client) vers Amazon. CloudWatch Lorsque des journaux côté appareil sont publiés sur une rubrique MQTT, l'action Règles des CloudWatch journaux transfère les messages vers les journaux. CloudWatch Ce processus explique comment télécharger les journaux des appareils par lots à l'aide du `batchMode` paramètre d'action des règles activé (défini sur`true`).

Pour commencer à télécharger les journaux côté appareil vers CloudWatch, remplissez les conditions préalables suivantes.

Prérequis

Avant de commencer, vous devez exécuter les actions suivantes :

- Créez au moins un appareil IoT cible enregistré en AWS IoT Core tant qu' AWS IoT objet. Pour plus d'informations, consultez Create a thing object.
- Déterminez l'espace thématique MQTT pour l'ingestion et les erreurs. Pour plus d'informations sur les sujets MQTT et les conventions de dénomination recommandées, consultez la section sur les sujets Rubriques MQTT MQTT dans Importer des journaux côté appareil sur Amazon. CloudWatch

Pour plus d'informations sur ces conditions préalables, consultez la section Charger les journaux côté appareil vers. CloudWatch

Création d'un groupe de CloudWatch journaux

Pour créer un groupe de CloudWatch journaux, procédez comme suit. Choisissez l'onglet approprié selon que vous préférez effectuer les étapes via le AWS Management Console ou le AWS Command Line Interface (AWS CLI).

AWS Management Console

Pour créer un groupe de CloudWatch journaux à l'aide du AWS Management Console

1. Ouvrez le AWS Management Console et naviguez vers CloudWatch.
2. Dans le panneau de navigation, choisissez Logs, puis Log groups
3. Sélectionnez Créer un groupe de journaux.
4. Mettez à jour le nom du groupe de journaux et, éventellement, mettez à jour les champs des paramètres de rétention.
5. Choisissez Créer.

AWS CLI

Pour créer un groupe de CloudWatch journaux à l'aide du AWS CLI

1. Exécutez les commandes suivantes pour créer le groupe de journaux. Pour plus d'informations, consultez `create-log-group` le manuel de référence des commandes AWS CLI v2.

   Remplacez le nom du groupe de journaux dans l'exemple (`uploadLogsGroup`) par votre nom préféré.

   ```
   aws logs create-log-group --log-group-name uploadLogsGroup
   ```

2. Pour confirmer que le groupe de journaux a été créé correctement, exécutez la commande suivante.

   ```
   aws logs describe-log-groups --log-group-name-prefix uploadLogsGroup
   ```

   Exemple de sortie :

```
{
    "logGroups": [
        {
            "logGroupName": "uploadLogsGroup",
            "creationTime": 1674521804657,
            "metricFilterCount": 0,
            "arn": "arn:aws:logs:us-east-1:111122223333:log-
group:uploadLogsGroup:*",
            "storedBytes": 0
        }
    ]
}
```

Création d'une règle de rubrique

Pour créer une AWS IoT règle, procédez comme suit. Choisissez l'onglet approprié selon que vous préférez effectuer les étapes via le AWS Management Console ou le AWS Command Line Interface (AWS CLI).

AWS Management Console

Pour créer une règle de sujet à l'aide du AWS Management Console

1. Ouvrez le hub de règles.
   a. Ouvrez le AWS Management Console et naviguez vers AWS IoT.
   b. Dans la barre de navigation, choisissez Routage des messages, puis Règles.
   c. Choisissez Créer une règle.
2. Entrez les propriétés de la règle.
   a. Entrez un nom de règle alphanumérique.
   b. (Facultatif) Entrez une Règle de description puis Onglet.
   c. Choisissez Suivant.
3. Entrez une instruction SQL.
   a. Entrez une instruction SQL en utilisant la rubrique MQTT que vous avez définie pour l'ingestion.

Par exemple, `SELECT * FROM '$aws/rules/things/thing_name/logs'`

b. Choisissez Suivant.

4. Entrez les actions des règles.

a. Dans le menu Action 1, choisissez CloudWatchlogs.

b. Choisissez le nom de groupe de journal puis sélectionnez le groupe de journal que vous avez crée.

c. Sélectionnez Utiliser le mode batch.

d. Spécifiez le rôle IAM pour la règle.

Si vous avez un rôle IAM pour la règle, procédez comme suit.

1. Dans le menu du rôle IAM, choisissez votre rôle IAM.

Si vous n'avez pas de rôle IAM pour la règle, procédez comme suit.

1. Choisissez Create New Role (Créer un nouveau rôle).
2. Pour Nom du rôle, entrez un nom unique et choisissez Créer.
3. Vérifiez que le nom du rôle IAM est correct dans le champ du rôle IAM.

e. Choisissez Suivant.

5. Passez en revue la configuration du modèle.

a. Vérifiez les paramètres du modèle de Tâche pour vérifier qu'ils sont corrects.

b. Lorsque vous avez terminé, cliquez sur Create (Créer).

AWS CLI

Pour créer un rôle IAM et une règle de sujet à l'aide du AWS CLI

1. Créez un rôle IAM qui accorde des droits à la AWS IoT règle.

a. Créez une politique IAM.

Exécutez la commande de l' suivante pour créer une Politique IAM Assurez-vous de mettre à jour la `policy-name` valeur du paramètre. Pour plus d'informations, consultez `create-policy`le manuel de référence des commandes AWS CLI v2.

> **Note**
>
> Si vous utilisez un système d'exploitation Microsoft Windows, vous devrez peut-être remplacer le marqueur de fin de ligne (\) par une coche (`) ou un autre caractère.

```
aws iam create-policy \
    --policy-name uploadLogsPolicy \
    --policy-document \
'{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateTopicRule",
            "iot:Publish",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:GetLogEvents"
        ],
        "Resource": "*"
    }
}'
```

b. Copiez l'ARN de la politique de votre sortie dans un éditeur de texte.

Exemple de sortie :

```
{
    "Policy": {
        "PolicyName": "uploadLogsPolicy",
        "PermissionsBoundaryUsageCount": 0,
        "CreateDate": "2023-01-23T18:30:10Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "AAABBBCCCDDDEEEFFFGGG",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::111122223333:policy/uploadLogsPolicy",
```

```
        "UpdateDate": "2023-01-23T18:30:10Z"
    }
}
```

c. Créer un rôle IAM et une politique de confiance.

Exécutez la commande de l' suivante pour créer une politique IAM Assurez-vous de mettre à jour la `role-name` valeur du paramètre. Pour plus d'informations, consultez `create-role`le manuel de référence des commandes AWS CLI v2.

```
aws iam create-role \
--role-name uploadLogsRole \
--assume-role-policy-document \
'{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'
```

d. Attachez la politique IAM au rôle.

Exécutez la commande de l' suivante pour créer une politique IAM Assurez-vous de mettre à jour les valeurs `role-name` et `policy-arn` paramètres. Pour plus d'informations, consultez `attach-role-policy`le manuel de référence des commandes AWS CLI v2.

```
aws iam attach-role-policy \
--role-name uploadLogsRole \
--policy-arn arn:aws:iam::111122223333:policy/uploadLogsPolicy
```

e. Passez en revue le rôle.

Pour vérifier que le rôle IAM a été créé correctement, exécutez la commande suivante. Assurez-vous de mettre à jour la valeur du `role-name` paramètre. Pour plus

d'informations, consultez `get-role`le manuel de référence des commandes AWS CLI v2.

```
aws iam get-role --role-name uploadLogsRole
```

Exemple de sortie :

```
{
    "Role": {
        "Path": "/",
        "RoleName": "uploadLogsRole",
        "RoleId": "AAABBBCCCDDDEEEFFFGGG",
        "Arn": "arn:aws:iam::111122223333:role/uploadLogsRole",
        "CreateDate": "2023-01-23T19:17:15+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "iot.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}
```

2. Créez une règle de AWS IoT sujet dans le AWS CLI.

   a. Pour créer une règle de AWS IoT rubrique, exécutez la commande suivante. Assurez-vous de mettre à jour le `--rule-name`, l'`sql`instruction, `description`, `roleARN` , et des `logGroupName` valeur de paramètres. Pour plus d'informations, consultez create-topic-rulele manuel de référence des commandes AWS CLI v2.

   ```
   aws iot create-topic-rule \
   --rule-name uploadLogsRule \
   ```

```
--topic-rule-payload \
 '{
  "sql":"SELECT * FROM 'rules/things/thing_name/logs'",
  "description":"Upload logs test rule",
  "ruleDisabled":false,
  "awsIotSqlVersion":"2016-03-23",
  "actions":[
   {"cloudwatchLogs":
    {"roleArn":"arn:aws:iam::111122223333:role/uploadLogsRole",
    "logGroupName":"uploadLogsGroup",
    "batchMode":true}
    }
   ]
}'
```

b. Pour vérifier que la règle a été créée correctement, exécutez la commande suivante. Assurez-vous de mettre à jour la valeur du `role-name` paramètre. Pour plus d'informations, consultez get-topic-rulele manuel de référence des commandes AWS CLI v2.

```
aws iot get-topic-rule --rule-name uploadLogsRule
```

Exemple de sortie :

```
{
    "ruleArn": "arn:aws:iot:us-east-1:111122223333:rule/uploadLogsRule",
    "rule": {
        "ruleName": "uploadLogsRule",
        "sql": "SELECT * FROM rules/things/thing_name/logs",
        "description": "Upload logs test rule",
        "createdAt": "2023-01-24T16:28:15+00:00",
        "actions": [
            {
                "cloudwatchLogs": {
                    "roleArn": "arn:aws:iam::111122223333:role/
uploadLogsRole",
                    "logGroupName": "uploadLogsGroup",
                    "batchMode": true
                }
            }
        ],
        "ruleDisabled": false,
```

```
        "awsIotSqlVersion": "2016-03-23"
    }
}
```

Envoi des journaux côté appareil à AWS IoT

Pour envoyer les journaux côté appareil à AWS IoT

1. Pour envoyer des journaux historiques à AWS IoT, communiquez avec vos appareils pour vous assurer de ce qui suit.

   - Les informations du journal sont envoyées à l'espace de noms de rubrique approprié, comme indiqué dans la section Conditions préalables de cette procédure.

     Par exemple, `$aws/rules/things/thing_name/logs`

   - La charge utile du message MQTT est correctement formatée. Pour plus d'informations sur MQTT et la convention de dénomination recommandée, consultez la Rubriques MQTT section interne Téléchargez les journaux côté appareil sur Amazon CloudWatch

2. Vérifiez que les messages MQTT sont reçus dans le client AWS IoT MQTT.

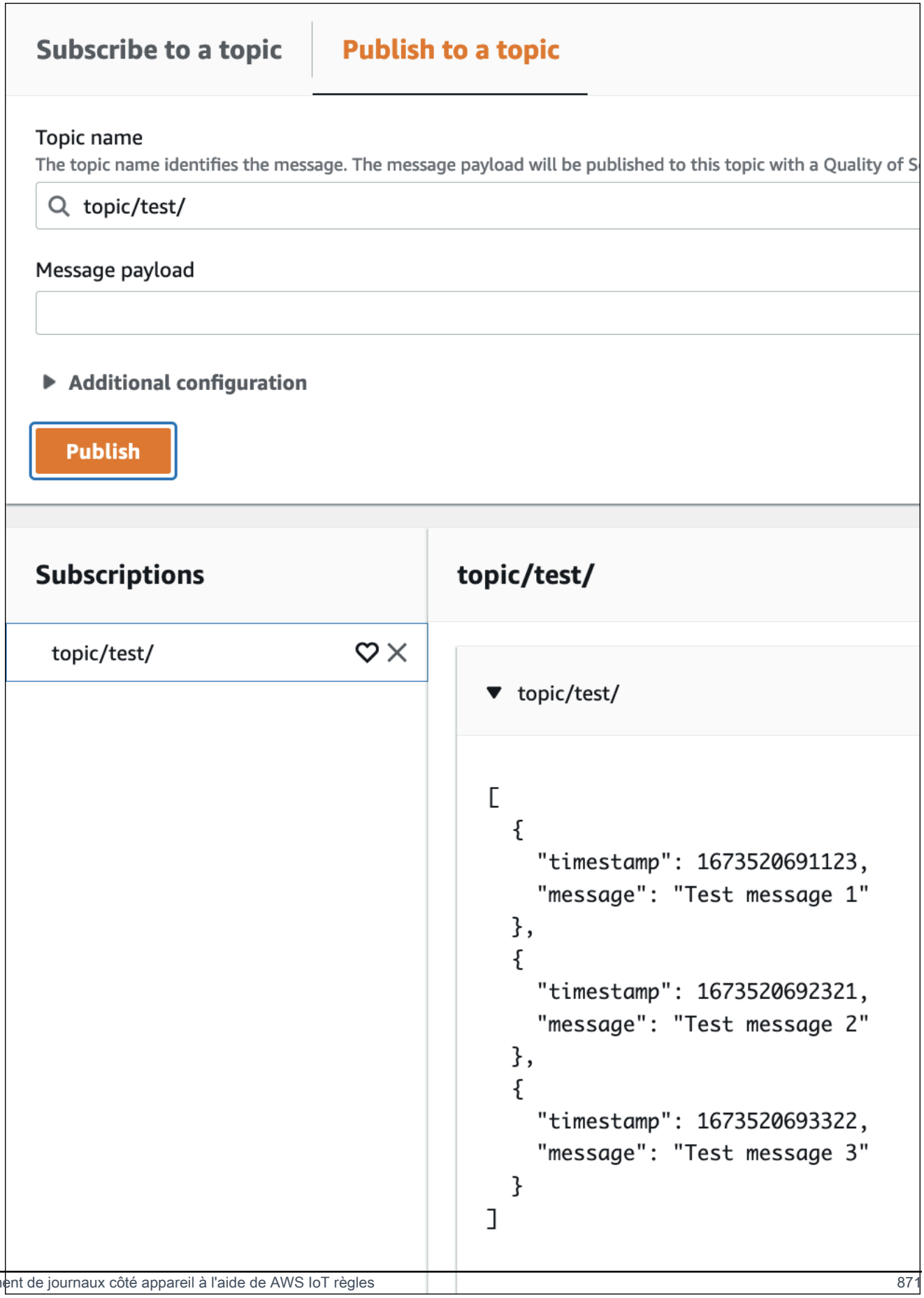
   a. Ouvrez le AWS Management Console et naviguez vers AWS IoT.

   b. Pour afficher le client de test MQTT, dans la barre de navigation, choisissez Test, client de test MQTT.

   c. Pour S'abonner à une rubrique, Filtre de rubrique, entrez l'espace de noms de rubrique.

   d. Choisissez Souscrire.

      Les messages MQTT apparaissent dans le tableau des abonnements table de Rubriques, comme indiqué ci-dessous. Ces messages peuvent mettre jusqu'à cinq minutes avant d'apparaître.


Subscribe to a topic
Publish to a topic
Topic name
The topic name identifies the message. The message payload will be published to this topic with a Quality of S
topic/test/
Message payload
Additional configuration
Publish
Subscriptions
topic/test/
topic/test/
topic/test/
[
{
"timestamp": 1673520691123,
"message": "Test message 1"
},
{
"timestamp": 1673520692321,
"message": "Test message 2"
},
{
"timestamp": 1673520693322,
"message": "Test message 3"
}
]

Afficher les données du journal

Pour consulter vos enregistrements de journal dans CloudWatch Logs

1. Ouvrez le AWS Management Console, puis naviguez vers CloudWatch.
2. Dans la barre de navigation, choisissez Logs, Logs Insights.
3. Dans le menu Sélectionner un ou plusieurs groupes de journaux, choisissez le groupe de journaux que vous avez spécifié dans la AWS IoT règle.
4. Sur la page Logs Insights, choisissez Run query.

Enregistrement des appels AWS IoT d'API à l'aide de AWS CloudTrail

AWS IoT est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans AWS IoT. CloudTrail capture tous les appels d'API AWS IoT sous forme d'événements, y compris les appels depuis la AWS IoT console et les appels de code vers le AWS IoT APIs. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrailévénements vers un compartiment Amazon S3, y compris les événements pour AWS IoT. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite AWS IoT, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et d'autres détails.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

AWS IoT informations dans CloudTrail

CloudTrail est activé sur votre compte Compte AWS lorsque vous créez le compte. Lorsqu'une activité se produit dans AWS IoT, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre Compte AWS. Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements dans votre Compte AWS, y compris les événements pour AWS IoT, créez un journal d'activité. Un suivi permet CloudTrail de fournir des fichiers journaux

à un compartiment Amazon S3. Par défaut, lorsque vous créez un parcours dans la console, celui-ci s'applique à tous les Région AWS s. Le journal enregistre les événements de tous les Région AWS s de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. Vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez :

- Présentation de la création d'un journal d'activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

> **Note**
>
> AWS IoT les actions du plan de données (côté appareil) ne sont pas enregistrées CloudTrail. CloudWatch À utiliser pour surveiller ces actions.

D'une manière générale, les actions du plan de AWS IoT contrôle qui apportent des modifications sont enregistrées par CloudTrail. Des appels tels que des entrées CreateThingCreateKeysAndCertificate, UpdateCertificateet des CloudTrail entrées, tandis que des appels tels ListTopicRulesque ListThingset non.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou IAM.
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

AWS IoT les actions sont documentées dans la référence de AWS IoT l'API. AWS IoT Wireless les actions sont documentées dans la référence de AWS IoT Wireless l'API.

Comprendre les entrées du fichier AWS IoT journal

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'`AttachPolicy`action.


```
{
    "timestamp":"1460159496",
    "AdditionalEventData":"",
    "Annotation":"",
    "ApiVersion":"",
    "ErrorCode":"",
    "ErrorMessage":"",
    "EventID":"8bff4fed-c229-4d2d-8264-4ab28a487505",
    "EventName":"AttachPolicy",
    "EventTime":"2016-04-08T23:51:36Z",
    "EventType":"AwsApiCall",
    "ReadOnly":"",
    "RecipientAccountList":"",
    "RequestID":"d4875df2-fde4-11e5-b829-23bf9b56cbcd",
    "RequestParamters":{
        "principal":"arn:aws:iot:us-
east-1:123456789012:cert/528ce36e8047f6a75ee51ab7beddb4eb268ad41d2ea881a10b67e8e76924d894",
        "policyName":"ExamplePolicyForIoT"
    },
    "Resources":"",
    "ResponseElements":"",
    "SourceIpAddress":"52.90.213.26",
    "UserAgent":"aws-internal/3",
    "UserIdentity":{
        "type":"AssumedRole",
        "principalId":"AKIAI44QH8DHBEXAMPLE",
        "arn":"arn:aws:sts::12345678912:assumed-role/iotmonitor-us-east-1-beta-
InstanceRole-1C5T1YCYMHPYT/i-35d0a4b6",
        "accountId":"222222222222",
```

```
        "accessKeyId":"access-key-id",
        "sessionContext":{
            "attributes":{
                "mfaAuthenticated":"false",
                "creationDate":"Fri Apr 08 23:51:10 UTC 2016"
            },
            "sessionIssuer":{
                "type":"Role",
                "principalId":"AKIAI44QH8DHBEXAMPLE",
                "arn":"arn:aws:iam::123456789012:role/executionServiceEC2Role/
iotmonitor-us-east-1-beta-InstanceRole-1C5T1YCYMHPYT",
                "accountId":"222222222222",
                "userName":"iotmonitor-us-east-1-InstanceRole-1C5T1YCYMHPYT"
            }
        },
        "invokedBy":{
            "serviceAccountId":"111111111111"
        }
    },
    "VpcEndpointId":""
}
```

Règles pour AWS IoT

Les règles permettent à vos appareils d'interagir avec Services AWS. Les règles sont analysées et les actions exécutées en fonction du flux de rubrique MQTT. Vous pouvez utiliser des règles pour effectuer les tâches suivantes :

- Augmenter ou filtrer les données reçues d'un appareil.
- Écrire les données reçues d'un appareil dans une base de données Amazon DynamoDB.
- Enregistrer un fichier dans Amazon S3.
- Envoyer une notification push à tous les utilisateurs qui utilisent Amazon SNS
- Publier des donnés dans une file d'attente Amazon SQS.
- Invoquer une fonction Lambda pour extraire des données.
- Traiter les messages provenant d'un grand nombre d'appareils à l'aide de Amazon Kinesis.
- Envoyez des données à Amazon OpenSearch Service.
- Capturez une CloudWatch métrique.
- Changez une CloudWatch alarme.
- Envoyez les données d'un message MQTT à Amazon SageMaker AI pour établir des prédictions basées sur un modèle d'apprentissage automatique (ML).
- Envoyer un message à un flux d'entrée Salesforce IoT
- Envoyez des données de message à un AWS IoT Analytics canal.
- Processus de démarrage d'un automate à fonctions par étapes.
- Envoyer les données du message à une AWS IoT Events entrée.
- Envoyer des données de message à une propriété de l'actif dans AWS IoT SiteWise.
- Envoyer des données de message à une application web ou à un service.

Vos règles peuvent utiliser des messages MQTT qui passent par le publish/subscribe protocole pris en charge par lethe section called “Protocoles de communication des appareils”. Vous pouvez également utiliser la fonction Basic Ingest pour envoyer en toute sécurité les données de l'appareil aux adresses Services AWS répertoriées précédemment, sans frais de messagerie. La fonction d'ingestion de base optimise le flux de données en supprimant le courtier de publish/subscribe messages du chemin d'ingestion. Cela le rend rentable tout en conservant les fonctionnalités de sécurité et de traitement des données de AWS IoT.

Avant de AWS IoT pouvoir effectuer ces actions, vous devez lui accorder l'autorisation d'accéder à vos AWS ressources en votre nom. Lorsque les actions sont effectuées, vous devez payer les frais standard pour celles Services AWS que vous utilisez.

Table des matières

- Accorder à une AWS IoT règle l'accès dont elle a besoin
- Transmission des autorisations de rôle
- Création d'une AWS IoT règle
- Gérer une AWS IoT règle
- AWS IoT actions liées aux règles
- Résolution des problèmes d'une règle
- Accès aux ressources entre comptes à l'aide AWS IoT de règles
- Gestion des erreurs (action d'erreur)
- Réduction des coûts de messagerie avec Basic Ingest
- AWS IoT Référence SQL

Accorder à une AWS IoT règle l'accès dont elle a besoin

Utilisez les rôles IAM pour contrôler les AWS ressources auxquelles chaque règle a accès. Avant de créer une règle, vous devez créer un rôle IAM avec une politique qui autorise l'accès aux AWS ressources requises. AWS IoT assume ce rôle lors de la mise en œuvre d'une règle.

Procédez comme suit pour créer le rôle et la AWS IoT politique IAM qui accordent à une AWS IoT règle l'accès dont elle a besoin (AWS CLI).

1. Enregistrez le document de politique de confiance suivant, qui accorde AWS IoT l'autorisation d'assumer le rôle, dans un fichier nomméiot-role-trust.json.

   JSON

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

```
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:iot:us-
east-1:123456789012:rule/rulename"
                }
            }
        }
    ]
}
```

Utilisez la commande create-role pour créer un rôle IAM spécifiant le fichier `iot-role-trust.json` :

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document
 file://iot-role-trust.json
```

La sortie de cette commande ressemble à ce qui suit :

```
{
 "Role": {
  "AssumeRolePolicyDocument": "url-encoded-json",
  "RoleId": "AKIAIOSFODNN7EXAMPLE",
  "CreateDate": "2015-09-30T18:43:32.821Z",
  "RoleName": "my-iot-role",
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:role/my-iot-role"
 }
}
```

2. Enregistrez le code JSON suivant dans un fichier nommé `my-iot-policy.json`.

   JSON

```
{
```

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

Ce JSON est un exemple de document de politique qui accorde à AWS IoT l'administrateur l'accès à DynamoDB.

Utilisez la commande create-policy pour autoriser l' AWS IoT accès à vos AWS ressources lorsque vous assumez le rôle, en transmettant le `my-iot-policy.json` fichier :

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy.json
```

Pour plus d'informations sur la manière d'accorder l'accès à Services AWS dans les politiques pour AWS IoT, voirCréation d'une AWS IoT règle.

La sortie de la commande create-policy contient l'ARN de la stratégie. Attacher la politique à un rôle.

```
{
 "Policy": {
  "PolicyName": "my-iot-policy",
  "CreateDate": "2015-09-30T19:31:18.620Z",
  "AttachmentCount": 0,
  "IsAttachable": true,
  "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
  "DefaultVersionId": "v1",
  "Path": "/",
  "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy",
  "UpdateDate": "2015-09-30T19:31:18.620Z"
 }
}
```

3. Utilisez la attach-role-policycommande pour associer votre politique à votre rôle :

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn
 "arn:aws:iam::123456789012:policy/my-iot-policy"
```

Révoquer l'accès au moteur de règles

Pour révoquer immédiatement l'accès au moteur de règles, procédez comme suit

1. Supprimer iot.amazonaws.com de la politique de confiance
2. Suivez les étapes pour révoquer les sessions IoT Role

Transmission des autorisations de rôle

Une définition de règle comporte un rôle IAM qui accorde l'autorisation d'accéder aux ressources spécifiées dans l'action de la règle. Le moteur de règles assume ce rôle lorsque l'action de la règle est invoquée.. Le rôle doit être défini de la même manière Compte AWS que la règle.

Lorsque vous créez ou remplacez une règle, vous transférez en fait un rôle au moteur de règles. L'`iam:PassRole`autorisation est requise pour effectuer cette opération. Pour vous assurer que vous disposez de cette autorisation, créez une politique qui accorde `iam:PassRole` l'autorisation et attachez-la à votre utilisateur IAM. La stratégie suivante montre comment accorder l'autorisation `iam:PassRole` pour un rôle.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Stmt1",
   "Effect": "Allow",
   "Action": [
    "iam:PassRole"
   ],
   "Resource": [
    "arn:aws:iam::123456789012:role/myRole"
   ]
  }
```

```
  ]
}
```

Dans cet exemple de politique, `iam:PassRole` l'autorisation est accordée pour le rôle `myRole`. Le rôle a été spécifié à l'aide de son ARN. Attacher cette politique à votre utilisateur IAM ou au rôle auquel appartient votre utilisateur. Pour plus d'informations, consultez Utilisation des politiques gérées.

> **Note**
>
> Les fonctions Lambda utilisent des politiques basées sur des ressources, où la politique est attachée directement à la fonction Lambda. Lorsque vous créez une règle qui appelle une fonction Lambda, vous ne transmettez pas de rôle, et l'utilisateur qui crée la règle n'a donc pas besoin de `iam:PassRole` l'autorisation. Pour plus d'informations sur l'autorisation de fonctions , consultez Octroi de permissions à l'aide d'une politique de ressources.

Création d'une AWS IoT règle

Vous pouvez créer des AWS IoT règles pour acheminer les données de vos objets connectés afin d'interagir avec d'autres AWS services. Une AWS IoT règle comprend les éléments suivants :

Composantes d'une règle

Composant	Description	Obligatoire ou facultatif
Nom de la règle	Le nom de la règle . Notez que nous ne recommandons pas l'utilisation d'informations personnellement identifiables dans les noms de vos règles.	Obligatoire.
Description de la règle	Description textuelle de la règle. Notez que nous ne recommandons pas l'utilisation d'informations personnellement identifiables dans les descriptions de vos règles.	Facultatif.
Instruction SQL	Syntaxe SQL simplifiée pour filtrer les messages reçus dans une rubrique MQTT et pousser les	Obligatoire.

Composant	Description	Obligatoire ou facultatif
	données ailleurs. Pour de plus amples informati ons, veuillez consulter AWS IoT Référence SQL.	
Version de SQL	Version du moteur de règles SQL à utiliser lors de l'évaluation de la règle. Même si cette propriété est facultative, nous vous recommandons vivement de préciser la version de SQL. La AWS IoT Core console définit cette propriété sur `2016-03-23` par défaut. Si cette propriété n'est pas définie, par exemple dans une AWS CLI commande ou un AWS CloudFormation modèle, elle `2015-10-0 8` est utilisée. Pour de plus amples informations, veuillez consulter Versions de SQL.	Obligatoire.
Une ou plusieurs actions	Les actions sont effectuées AWS IoT lors de la mise en œuvre de la règle. Par exemple, vous pouvez insérer des données dans un tableau DynamoDB, écrire des données dans un compartiment Amazon S3, publier dans une rubrique Amazon SNS ou appeler une fonction Lambda.	Obligatoire.
Une action d'erreur	L'action est AWS IoT exécutée lorsqu'elle n'est pas en mesure d'exécuter l'action d'une règle.	Facultatif.

Avant de créer une AWS IoT règle, vous devez créer un rôle IAM avec une politique qui autorise l'accès aux AWS ressources requises. AWS IoT assume ce rôle lors de la mise en œuvre d'une règle. Pour plus d'informations, consultez les sections Accorder à une AWS IoT règle l'accès dont elle a besoin et Transmission des autorisations de rôle.

Lorsque vous créez une règle, soyez conscient de la quantité de données que vous publiez sur les sujets. Si vous créez des règles qui incluent un modèle de sujet générique, elles peuvent correspondre à un pourcentage important de vos messages. Dans ce cas, vous devrez peut-être augmenter la capacité des AWS ressources utilisées par les actions cibles. Nous recommandons

d'éviter les modèles de sujets génériques dans les règles de republication afin d'éviter le double traitement et de réduire les coûts.

> **Note**
>
> La création et la mise à jour de règles sont des actions de niveau administrateur. Tout utilisateur détenant des autorisations de création ou de mise à jour de règles peut accéder aux données traitées par les règles.

Création d'une règle (console)

Pour créer une règle (AWS Management Console)

Utilisez la AWS Management Consolecommande pour créer une règle :

1. Ouvrez la AWS IoT console.
2. Dans le volet de navigation de gauche, choisissez Routage des messages dans la section Gérer. Choisissez ensuite Règles.
3. Sur la page Règles, choisissez Créer une règle.
4. Sur la page Spécifier les propriétés de la règle, entrez le nom de votre règle. La description des règles et les balises sont facultatives. Choisissez Suivant.
5. Sur la page Configurer l'instruction SQL, choisissez une version SQL et entrez une instruction SQL. Un exemple d'instruction SQL peut êtreSELECT temperature FROM 'iot/topic' WHERE temperature > 50. Pour plus d'informations, voir Versions SQL et références AWS IoT SQL.
6. Sur la page Joindre des actions de règle, ajoutez des actions de règle pour acheminer les données vers d'autres AWS services.

    1. Dans Actions de règle, sélectionnez une action de règle dans la liste déroulante. Par exemple, vous pouvez choisir Kinesis Stream. Pour plus d'informations sur les actions des règles, consultez la section Actions des AWS IoT règles.
    2. En fonction de l'action de règle que vous choisissez, entrez les détails de configuration associés. Par exemple, si vous choisissez Kinesis Stream, vous devrez choisir ou créer une ressource de flux de données, et éventuellement saisir les détails de configuration tels que la clé de partition, qui est utilisée pour regrouper les données par partition dans un Steam.

3. Dans le rôle IAM, choisissez ou créez un rôle pour accorder l' AWS IoT accès à votre point de terminaison. Notez que cela AWS IoT créera automatiquement une politique avec le préfixe « `aws-iot-rule` sous le rôle IAM sélectionné ». Vous pouvez choisir Afficher pour afficher votre rôle IAM et la politique depuis la console IAM. L'action en cas d'erreur est facultative. Vous trouverez plus d'informations dans Gestion des erreurs (action en cas d'erreur). Pour plus d'informations sur la création d'un rôle IAM pour votre règle, voir Accorder à une règle l'accès dont elle a besoin. Choisissez Suivant.

7. Sur la page Réviser et créer, passez en revue l'ensemble de la configuration et apportez des modifications si nécessaire. Choisissez Créer.

Une fois que vous avez créé une règle avec succès, elle apparaît sur la page Règles. Vous pouvez sélectionner une règle pour ouvrir la page Détails où vous pouvez afficher une règle, modifier une règle, désactiver une règle et supprimer une règle.

Création d'une règle (CLI)

Pour créer une règle (AWS CLI)

Utilisez la create-topic-rulecommande pour créer une règle :

```
aws iot create-topic-rule --rule-name myrule --topic-rule-payload file://myrule.json
```

Voici un exemple de fichier de charge utile avec une règle qui insère tous les messages envoyés à la `iot/test` rubrique dans le tableau DynamoDB spécifiée. L'instruction SQL filtre les messages et le rôle ARN AWS IoT autorise l'écriture dans la table DynamoDB.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "dynamoDB": {
    "tableName": "my-dynamodb-table",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "hashKeyField": "topic",
    "hashKeyValue": "${topic(2)}",
    "rangeKeyField": "timestamp",
    "rangeKeyValue": "${timestamp()}"
```

```
    }
   }
 ]
}
```

Voici un exemple de fichier de charge utile avec une règle qui insère tous les messages envoyés à la rubrique `iot/test` dans le compartiment S3 spécifié. L'instruction SQL filtre les messages, et l'ARN du rôle accorde AWS IoT l'autorisation d'écrire dans le compartiment Amazon S3.

```
{
 "awsIotSqlVersion": "2016-03-23",
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "actions": [
  {
   "s3": {
    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
    "bucketName": "amzn-s3-demo-bucket",
    "key": "myS3Key"
   }
  }
 ]
}
```

Voici un exemple de fichier de charge utile avec une règle qui envoie des données vers Amazon OpenSearch Service :

```
{
 "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "OpenSearch": {
    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es",
    "endpoint": "https://my-endpoint",
    "index": "my-index",
    "type": "my-type",
    "id": "${newuuid()}"
   }
  }
 ]
```

```
}
```

Voici un exemple de fichier de charge avec une règle qui appelle une fonction Lambda :

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "lambda": {
    "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function"
   }
  }
 ]
}
```

Voici un exemple de fichier de charge utile avec une règle qui publie dans une rubrique Amazon SNS :

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sns": {
    "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Voici un exemple de fichier de charge utile avec une règle qui permet de republier dans une autre rubrique MQTT :

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
```

```
  {
   "republish": {
    "topic": "my-mqtt-topic",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Voici un exemple de fichier de charge utile avec une règle qui envoie des données vers un flux Amazon Data Firehose :

```
{
 "sql": "SELECT * FROM 'my-topic'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "firehose": {
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "deliveryStreamName": "my-stream-name"
   }
  }
 ]
}
```

Voici un exemple de fichier de charge utile avec une règle qui utilise la `machinelearning_predict` fonction Amazon SageMaker AI pour republier dans une rubrique si les données de la charge utile MQTT sont classées comme 1.

```
{
 "sql": "SELECT * FROM 'iot/test' where machinelearning_predict('my-model',
 'arn:aws:iam::123456789012:role/my-iot-aml-role', *).predictedLabel=1",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "republish": {
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
    "topic": "my-mqtt-topic"
   }
  }
```

```
 ]
}
```

Voici un exemple de fichier de charge utile assorti d'une règle qui publie des messages dans un flux d'entrée Salesforce IoT Cloud.

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "salesforce": {
    "token": "ABCDEFGHI123456789abcdefghi123456789",
    "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/
connection-id/my-event"
   }
  }
 ]
}
```

L'exemple suivant illustre un fichier de charge utile avec une règle lançant l'exécution d'une machine d'état Step Functions.

```
{
 "sql": "expression",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "stepFunctions": {
    "stateMachineName": "myCoolStateMachine",
    "executionNamePrefix": "coolRunning",
    "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
   }
  }
 ]
}
```

Gérer une AWS IoT règle

Vous pouvez utiliser les actions suivantes pour gérer vos AWS IoT règles.

Dans cette rubrique :

- Marquer une règle
- Afficher une règle
- Suppression d'une règle

Marquer une règle

Pour ajouter un niveau de spécificité à vos règles nouvelles ou existantes, vous pouvez appliquer le balisage. Le balisage utilise des paires clé-valeur dans vos règles pour vous permettre de mieux contrôler comment et où vos règles sont appliquées à vos AWS IoT ressources et services. Par exemple, vous pouvez limiter le champ d'application de votre règle pour qu'elle s'applique uniquement à votre environnement bêta pour les tests préliminaires (`Key=environment, Value=beta`) ou pour capturer tous les messages envoyés au `iot/test` sujet depuis un point de terminaison spécifique uniquement et les stocker dans un compartiment Amazon S3.

Exemple de politique IAM

Pour un exemple qui montre comment accorder des autorisations de balisage pour une règle, imaginez un utilisateur qui exécute la commande suivante pour créer une règle et la baliser afin qu'elle ne s'applique qu'à son environnement bêta.

Dans l'exemple, remplacez :

- *`MyTopicRuleName`*avec le nom de la règle.
- *`myrule.json`*avec le nom du document de politique.

```
aws iot create-topic-rule
    --rule-name MyTopicRuleName
    --topic-rule-payload file://myrule.json
    --tags "environment=beta"
```

Pour cet exemple, vous devez utiliser la politique IAM suivante :

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": [ "iot:CreateTopicRule", "iot:TagResource" ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:iot:us-east-1:123456789012:rule/MyTopicRuleName"
        ]
    }
}
```

L'exemple ci-dessus montre une nouvelle règle appelée `MyTopicRuleName` qui s'applique uniquement à votre environnement bêta. La `iot:TagResource` dans la déclaration de politique avec `MyTopicRuleName` spécifiquement appelé permet le marquage lors de la création ou de la mise à jour `MyTopicRuleName`. Le paramètre `--tags "environment=beta"` utilisé lors de la création de la règle limite le champ d'application `MyTopicRuleName` à votre environnement bêta uniquement. Si vous supprimez le paramètre `--tags "environment=beta"`, ensuite `MyTopicRuleName` s'appliquera à tous les environnements.

Pour plus d'informations sur la création de rôles et de politiques IAM spécifiques à une AWS IoT règle, voir Accorder à une AWS IoT règle l'accès dont elle a besoin

Pour obtenir des informations générales sur le balisage des ressources, veuillez consulter . Marquer vos ressources AWS IoT.

Afficher une règle

Utilisez la list-topic-rulescommande pour répertorier vos règles :

```
aws iot list-topic-rules
```

Utilisez la get-topic-rulecommande pour obtenir des informations sur une règle :

```
aws iot get-topic-rule --rule-name myrule
```

Suppression d'une règle

Lorsque vous avez terminé avec une règle, vous pouvez la supprimer.

Pour supprimer une règle (AWS CLI)

Utilisez la delete-topic-rulecommande pour supprimer une règle :

```
aws iot delete-topic-rule --rule-name myrule
```

AWS IoT actions liées aux règles

AWS IoT les actions de règle spécifient ce qu'il faut faire lorsqu'une règle est invoquée. Vous pouvez définir des actions pour envoyer des données vers une base de données Amazon DynamoDB, envoyer des données vers Amazon Kinesis Data Streams, AWS Lambda invoquer une fonction, etc. AWS IoT prend en charge les actions suivantes Régions AWS lorsque le service de l'action est disponible.

Action de la règle	Description	Nom dans l'API
Apache Kafka	Envoie un message à un cluster Apache Kafka.	`kafka`
CloudWatch alarmes	Modifie l'état d'une CloudWatch alarme Amazon.	`cloudwatchAlarm`
CloudWatch Journaux	Envoie un message à Amazon CloudWatch Logs.	`cloudwatchLogs`
CloudWatch métriques	Envoie un message à une CloudWatch métrique.	`cloudwatchMetric`
DynamoDB	Envoie un message à une table DynamoDB.	`dynamoDB`
Dynamo DBv2	Envoie les données des messages à plusieurs colonnes d'une table DynamoDB.	`dynamoDBv2`

Action de la règle	Description	Nom dans l'API
Elasticsearch	Envoie un message à un OpenSearch point de terminaison.	`OpenSearch`
HTTP	Publie un message sur un point de terminaison HTTPS.	`http`
IoT Analytics	Envoie un message à une AWS IoT Analytics chaîne.	`iotAnalytics`
AWS IoT Events	Envoie un message à une AWS IoT Events entrée.	`iotEvents`
AWS IoT SiteWise	Envoie les données des messages aux propriétés des AWS IoT SiteWise actifs.	`iotSiteWise`
Firehose	Envoie un message à un flux de diffusion Firehose.	`firehose`
Kinesis Data Streams	Envoie un message à un flux de données Kinesis.	`kinesis`
Lambda	Invoque une fonction Lambda avec des données de message en entrée.	`lambda`
Emplacement	Envoie les données de localisation au service de localisation d'Amazon.	`location`
OpenSearch	Envoie un message à un point de terminaison Amazon OpenSearch Service.	`OpenSearch`
Republish	Republie un message vers un autre sujet MQTT.	`republish`

Action de la règle	Description	Nom dans l'API
S3	Stocke un message dans un panier Amazon Simple Storage Service (Amazon S3).	`s3`
Salesforce IoT	Envoie un message à un flux d'entrée Salesforce IoT.	`salesforce`
SNS	Publie un message sous la forme d'une notification push Amazon Simple Notification Service (Amazon SNS).	`sns`
SQS	Envoie un message à une file d'attente Amazon Simple Queue Service (Amazon SQS).	`sqs`
Step Functions	Démarre une machine à AWS Step Functions états.	`stepFunctions`
the section called “Timestream”	Envoie un message à une table de base de données Amazon Timestream.	`timestream`

Remarques

- Définissez la règle de la même manière Région AWS que la ressource d'un autre service afin que l'action de la règle puisse interagir avec cette ressource.
- Le moteur de AWS IoT règles peut effectuer plusieurs tentatives pour exécuter une action si des erreurs intermittentes se produisent. Si toutes les tentatives échouent, le message est ignoré et l'erreur est répertoriée dans vos CloudWatch journaux. Vous pouvez spécifier une action en cas d'erreur pour chaque règle appelée après une défaillance. Pour de plus amples informations, veuillez consulter Gestion des erreurs (action d'erreur).

- Certaines actions de règles activent des actions dans des services qui s'intègrent à AWS Key Management Service (AWS KMS) pour prendre en charge le cryptage des données au repos. Si vous utilisez une clé KMS gérée par le client AWS KMS key pour chiffrer les données au repos, le service doit être autorisé à utiliser la clé KMS au nom de l'appelant. Pour savoir comment gérer les autorisations pour votre clé KMS gérée par le client, consultez les rubriques relatives au cryptage des données dans le guide de service approprié. Pour de plus amples informations sur les clés KMS gérées par le client, consultez la section AWS Key Management Service concepts dans le AWS Key Management Service Guide du développeur.

Apache Kafka

L'action Apache Kafka (Kafka) envoie des messages directement à votre Amazon Managed Streaming for Apache Kafka (Amazon MSK), aux clusters Apache Kafka gérés par des fournisseurs tiers tels que Confluent Cloud ou aux clusters Apache Kafka autogérés. Grâce à l'action des règles de Kafka, vous pouvez acheminer vos données IoT vers des clusters Kafka. Cela vous permet de créer des pipelines de données à hautes performances à diverses fins, telles que l'analyse en continu, l'intégration des données, la visualisation et les applications professionnelles critiques.

> **Note**
>
> Cette rubrique suppose une bonne connaissance de la plateforme Apache Kafka et des concepts associés. Pour de plus amples informations sur Apache Kafka, veuillez consulter Apache Kafka. MSK Serverless n'est pas pris en charge. Les clusters MSK Serverless ne peuvent être créés que via l'authentification IAM, que l'action des règles d'Apache Kafka ne prend pas en charge actuellement. Pour plus d'informations sur la configuration AWS IoT Core avec Confluent, voir Tirer parti de Confluent et résoudre les problèmes AWS de gestion des appareils et des données IoT.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer d'exécuter les `ec2:DescribeSecurityGroups` opérations `ec2:CreateNetworkInterfaceec2:DescribeNetworkInterfaces,ec2:CreateNetworkInterf`

et. Ce rôle crée et gère des interfaces réseau élastiques vers votre Amazon Virtual Private Cloud afin d'atteindre votre courtier Kafka. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT Core de cette action de règle.

Pour plus d'informations sur les interfaces réseau, consultez la section Elastic network interfaces dans le guide de EC2 l'utilisateur Amazon.

La stratégie associée au rôle que vous spécifiez doit ressembler à l'exemple suivant.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateNetworkInterface",
            "ec2:DescribeNetworkInterfaces",
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterface",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeVpcAttribute",
            "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

- Si vous stockez AWS Secrets Manager les informations d'identification requises pour vous connecter à votre courtier Kafka, vous devez créer un rôle IAM AWS IoT Core capable d'effectuer les opérations `secretsmanager:GetSecretValue` et`secretsmanager:DescribeSecret`.

  La stratégie associée au rôle que vous spécifiez doit ressembler à l'exemple suivant.

  JSON

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue",
            "secretsmanager:DescribeSecret"
            ],
            "Resource": [

 "arn:aws:secretsmanager:region:123456789012:secret:kafka_client_truststore-*",

 "arn:aws:secretsmanager:region:123456789012:secret:kafka_keytab-*"
            ]
        }
    ]
}
```

- Vous pouvez exécuter vos clusters Apache Kafka dans Amazon Virtual Private Cloud (Amazon VPC). Vous devez créer une destination Amazon VPC et utiliser une passerelle NAT dans vos sous-réseaux pour transférer les messages depuis un cluster AWS IoT Kafka public. Le moteur de AWS IoT règles crée une interface réseau dans chacun des sous-réseaux répertoriés dans la destination du VPC pour acheminer le trafic directement vers le VPC. Lorsque vous créez une destination VPC, le moteur de AWS IoT règles crée automatiquement une action de règle VPC. Pour plus d'informations sur les actions de règle VPC, consultez Destinations de cloud privé virtuel (VPC).
- Si vous utilisez une clé KMS gérée par AWS KMS key le client pour chiffrer les données au repos, le service doit être autorisé à utiliser la clé KMS au nom de l'appelant. Pour plus d'informations, consultez Actions Amazon MSK Kafka dans le Guide du développeur Amazon Managed Streaming for Apache Kafka.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

DestinationArn

ARN (Amazon Resource Name) de la destination VPC. Pour plus d'informations sur la création d'une destination VPC, consultez Destinations de cloud privé virtuel (VPC).

topic

La rubrique Kafka pour les messages à envoyer à l'agent Kafka.

Vous pouvez remplacer ce champ à l'aide d'un modèle de substitution. Pour de plus amples informations, veuillez consulter the section called “Modèles de substitution”.

clé (facultatif)

La clé de message Kafka.

Vous pouvez remplacer ce champ à l'aide d'un modèle de substitution. Pour de plus amples informations, veuillez consulter the section called “Modèles de substitution”.

en-têtes (facultatif)

La liste des en-têtes Kafka que vous spécifiez. Chaque en-tête est une paire clé-valeur que vous pouvez spécifier lors de la création d'une action Kafka. Vous pouvez utiliser ces en-têtes pour acheminer les données des clients IoT vers les clusters Kafka en aval sans modifier la charge utile de vos messages.

Vous pouvez remplacer ce champ à l'aide d'un modèle de substitution. Pour comprendre comment transmettre la fonction d'une règle intégrée en tant que modèle de substitution dans l'en-tête de Kafka Action, consultez les exemples. Pour de plus amples informations, veuillez consulter the section called “Modèles de substitution”.

> **Note**
>
> Les en-têtes au format binaire ne sont pas pris en charge.

partition (facultatif)

La partition de message Kafka.

Vous pouvez remplacer ce champ à l'aide d'un modèle de substitution. Pour de plus amples informations, veuillez consulter the section called “Modèles de substitution”.

ClientProperties

Un objet qui définit les propriétés du client producteur Apache Kafka.

Packs (facultatif)

Le nombre d'accusés de réception que le producteur demande au serveur de recevoir avant de considérer qu'une demande est complète.

Si vous spécifiez 0 comme valeur, le producteur n'attendra aucun accusé de réception du serveur. Si le serveur ne reçoit pas le message, le producteur ne réessaiera pas de l'envoyer.

Valeurs valides: `-1`, `0`, `1`, `all`. La valeur par défaut est `1`.

serveurs bootstrap

Liste des paires d'hôtes et de ports (par exemple`host1:port1`,`host2:port2`) utilisées pour établir la connexion initiale à votre cluster Kafka.

compression.type (facultatif)

Type de compression pour toutes les données générées par le producteur.

Valeurs valides: `none`, `gzip`, `snappy`, `lz4`, `zstd`. La valeur par défaut est `none`.

security.protocol

Le protocole de sécurité utilisé pour vous connecter à votre courtier Kafka.

Valeurs valides : `SSL`, `SASL_SSL`. La valeur par défaut est `SSL`.

key.serializer

Spécifie comment transformer en octets les objets clés que vous fournissez avec le`ProducerRecord`.

Valeur valide : `StringSerializer`.

value.serializer

Spécifie comment transformer en octets les objets de valeur que vous fournissez avec le`ProducerRecord`.

Valeur valide : `ByteBufferSerializer`.

ssl.truststore

Le fichier truststore au format base64 ou l'emplacement du fichier truststore dans. AWS Secrets Manager Cette valeur n'est pas obligatoire si votre truststore est approuvé par les autorités de certification Amazon (CA).

Ce champ prend en charge les modèles de substitution. Si vous utilisez Secrets Manager pour stocker les informations d'identification requises pour vous connecter à votre broker Kafka, vous pouvez utiliser la fonction `get_secret` SQL pour récupérer la valeur de ce champ. Pour de plus amples informations sur les modèles de substitution, veuillez consulter the section called “Modèles de substitution”. Pour plus d'informations sur la `get_secret` fonction SQL, veillez consultez the section called “get_secret (SecreTid, SecretType, clé, roLearn)”. Si le truststore se présente sous la forme d'un fichier, utilisez le `SecretBinary` paramètre. Si le truststore se présente sous la forme d'une chaîne, utilisez le `SecretString` paramètre.

La taille maximale de cette valeur est de 65 Ko.

ssl.truststore.password

Le mot de passe du référentiel d'approbations. Cette valeur n'est requise que si vous avez créé un mot de passe pour le référentiel d'approbations.

ssl.keystore

Le fichier keystore. Cette valeur est obligatoire lorsque vous spécifiez `SSL` comme valeur pour`security.protocol`.

Ce champ prend en charge les modèles de substitution. Utilisez Secrets Manager pour stocker les informations d'identification requises pour vous connecter à votre courtier Kafka. Pour récupérer la valeur de ce champ, utilisez la fonction `get_secret` SQL. Pour de plus amples informations sur les modèles de substitution, veuillez consulter the section called “Modèles de substitution”. Pour plus d'informations sur la fonction `get_secret` SQL, consultez the section called “get_secret (SecreTid, SecretType, clé, roLearn)”. Utilisez le `SecretBinary` paramètre.

ssl.keystore.password

Le mot de passe du fichier keystore. Cette valeur est requise si vous spécifiez une valeur pour `ssl.keystore`.

La valeur de ce champ peut être en texte brut. Ce champ prend également en charge les modèles de substitution. Utilisez Secrets Manager pour stocker les informations d'identification

requises pour vous connecter à votre courtier Kafka. Pour récupérer la valeur de ce champ, utilisez la fonction `get_secret` SQL. Pour de plus amples informations sur les modèles de substitution, veuillez consulter the section called “Modèles de substitution”. Pour plus d'informations sur la fonction `get_secret` SQL, consultez the section called “get_secret (SecreTid, SecretType, clé, roLearn)”. Utilisez le `SecretString` paramètre.

clé ssl.mot de passe

Le mot de passe de la clé privée dans votre fichier keystore.

Ce champ prend en charge les modèles de substitution. Utilisez Secrets Manager pour stocker les informations d'identification requises pour vous connecter à votre courtier Kafka. Pour récupérer la valeur de ce champ, utilisez la fonction `get_secret` SQL. Pour de plus amples informations sur les modèles de substitution, veuillez consulter the section called “Modèles de substitution”. Pour plus d'informations sur la fonction `get_secret` SQL, consultez the section called “get_secret (SecreTid, SecretType, clé, roLearn)”. Utilisez le `SecretString` paramètre.

sasl.mechanism

Le mécanisme de sécurité utilisé pour se connecter à votre courtier Kafka. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour`security.protocol`.

Valeurs valides : `PLAIN`, `SCRAM-SHA-512`, `GSSAPI`.

> **Note**
>
> `SCRAM-SHA-512`est le seul mécanisme de sécurité pris en charge dans les régions cn-north-1, cn-northwest-1, -1 et -1. us-gov-east us-gov-west

sasl.plain.username

Le nom d'utilisateur utilisé pour récupérer la chaîne secrète depuis Secrets Manager. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `PLAIN` pour `sasl.mechanism`.

sasl.plain.password

Mot de passe utilisé pour récupérer la chaîne secrète depuis Secrets Manager. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `PLAIN` pour `sasl.mechanism`.

sasl.scram.nom d'utilisateur

Le nom d'utilisateur utilisé pour récupérer la chaîne secrète depuis Secrets Manager. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `SCRAM-SHA-512` pour `sasl.mechanism`.

sasl.scram.password

Mot de passe utilisé pour récupérer la chaîne secrète depuis Secrets Manager. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `SCRAM-SHA-512` pour`sasl.mechanism`.

sasl.kerberos.keytab

Le fichier keytab pour l'authentification Kerberos dans Secrets Manager. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `GSSAPI` pour`sasl.mechanism`.

Ce champ prend en charge les modèles de substitution. Utilisez Secrets Manager pour stocker les informations d'identification requises pour vous connecter à votre courtier Kafka. Pour récupérer la valeur de ce champ, utilisez la fonction `get_secret` SQL. Pour de plus amples informations sur les modèles de substitution, veuillez consulter the section called “Modèles de substitution”. Pour plus d'informations sur la fonction `get_secret` SQL, consultez the section called “get_secret (SecreTid, SecretType, clé, roLearn)”. Utilisez le `SecretBinary` paramètre.

sasl.kerberos.service.name

Nom principal Kerberos sous lequel Apache Kafka s'exécute. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `GSSAPI` pour `sasl.mechanism`.

sasl.kerberos.krb5.kdc

Le nom d'hôte du centre de distribution de clés (KDC) auquel votre client producteur Apache Kafka se connecte. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `GSSAPI` pour `sasl.mechanism`.

sasl.kerberos.krb5.realm

Domaine auquel votre client producteur Apache Kafka se connecte. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `GSSAPI` pour `sasl.mechanism`.

sasl.kerberos.principal

Identité Kerberos unique à laquelle Kerberos peut attribuer des tickets pour accéder aux services compatibles Kerberos. Cette valeur est obligatoire lorsque vous spécifiez `SASL_SSL` pour `security.protocol` et `GSSAPI` pour `sasl.mechanism`.

Exemples

L'exemple JSON suivant définit une action Apache Kafka dans une AWS IoT règle. L'exemple suivant transmet la fonction en ligne SourceIP () comme modèle de substitution dans l'en-tête Kafka Action.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "kafka": {
     "destinationArn": "arn:aws:iot:region:123456789012:ruledestination/vpc/
VPCDestinationARN",
     "topic": "TopicName",
     "clientProperties": {
      "bootstrap.servers": "kafka.com:9092",
      "security.protocol": "SASL_SSL",
      "ssl.truststore": "${get_secret('kafka_client_truststore',
 'SecretBinary','arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
      "ssl.truststore.password": "kafka password",
      "sasl.mechanism": "GSSAPI",
      "sasl.kerberos.service.name": "kafka",
      "sasl.kerberos.krb5.kdc": "kerberosdns.com",
      "sasl.kerberos.keytab": "${get_secret('kafka_keytab','SecretBinary',
 'arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
      "sasl.kerberos.krb5.realm": "KERBEROSREALM",
      "sasl.kerberos.principal": "kafka-keytab/kafka-keytab.com"
     },
     "headers": [
      {
       "key": "static_header_key",
       "value": "static_header_value"
      },
      {
```

```
        "key": "substitutable_header_key",
        "value": "${value_from_payload}"
       },
       {
        "key": "source_ip",
        "value": "${sourceIp()}"
       }
      ]
     }
    }
   ]
  }
}
```

Remarques importantes concernant votre configuration Kerberos

- Votre centre de distribution de clés (KDC) doit pouvoir être résolu via un système de noms de domaine (DNS) privé au sein de votre VPC cible. Une approche possible consiste à ajouter l'entrée DNS du KDC à une zone hébergée privée. Pour plus d'informations sur cette approche, veuillez consulter Utilisation des zones hébergées privées.
- Chaque VPC doit avoir la résolution DNS activée. Pour plus d'informations, consultez Utilisation de DNS avec votre VPC.
- Les groupes de sécurité de l'interface réseau et les groupes de sécurité au niveau de l'instance de destination du VPC doivent autoriser le trafic provenant de votre VPC sur les ports suivants.
  - Trafic TCP sur le port d'écoute du broker bootstrap (souvent 9092, mais doit être compris entre 9000 et 9100)
  - Trafic TCP et UDP sur le port 88 pour le KDC
- `SCRAM-SHA-512`est le seul mécanisme de sécurité pris en charge dans les régions cn-north-1, cn-northwest-1, -1 et -1. us-gov-east us-gov-west

Destinations de cloud privé virtuel (VPC)

L'action Apache Kafka achemine les données vers un cluster Apache Kafka dans un Amazon Virtual Private Cloud (Amazon VPC). La configuration VPC utilisée par l'action de règle Apache Kafka est automatiquement activée lorsque vous spécifiez la destination VPC pour votre action de règle.

Une destination VPC contient une liste de sous-réseaux à l'intérieur du VPC. Le moteur de règles crée une interface Elastic Network dans chaque sous-réseau que vous spécifiez dans cette liste.

Pour plus d'informations sur les interfaces réseau, consultez la section Elastic network interfaces dans le guide de EC2 l'utilisateur Amazon.

Exigences et considérations

- Si vous utilisez un cluster Apache Kafka autogéré accessible via un point de terminaison public sur Internet :
  - Créez une passerelle NAT pour les instances de vos sous-réseaux. La passerelle NAT possède une adresse IP publique qui peut se connecter à Internet, ce qui permet au moteur de règles de transférer vos messages au cluster Kafka public.
  - Allouez une adresse IP élastique avec les interfaces réseau élastiques (ENIs) créées par la destination du VPC. Les groupes de sécurité que vous utilisez doivent être configurés pour bloquer le trafic entrant.

  > **Note**
  >
  > Si la destination VPC est désactivée puis réactivée, vous devez réassocier l'Elastic IPs à la nouvelle. ENIs

- Si une destination de règle de sujet VPC ne reçoit aucun trafic pendant 30 jours consécutifs, elle sera désactivée.
- Si l'une des ressources utilisées par la destination du VPC change, la destination sera désactivée et ne pourra pas être utilisée.
- Parmi les modifications qui peuvent désactiver une destination VPC, citons : la suppression du VPC, des sous-réseaux, des groupes de sécurité ou du rôle utilisé ; la modification du rôle pour ne plus disposer des autorisations nécessaires ; et la désactivation de la destination.

Tarification

À des fins de tarification, une action de règle VPC est mesurée en plus de l'action qui envoie un message à une ressource lorsque celle-ci se trouve dans votre VPC. Pour en savoir plus sur la tarification, consultez Tarification AWS IoT Core.

Création de destinations de règle de rubrique Virtual Private Cloud (VPC)

Vous créez une destination de cloud privé virtuel (VPC) à l'aide de l'CreateTopicRuleDestinationAPI ou de la console. AWS IoT Core

When you create a VPC destination, you must specify the following information.

vpcId

L'identifiant unique de la destination VPC.

subnetIds

Liste de sous-réseaux dans lesquels le moteur de règles crée des interfaces réseau élastiques. Le moteur de règles alloue une interface réseau unique pour chaque sous-réseau de la liste.

SecurityGroups (facultatif)

Liste de groupes de sécurité à appliquer aux interfaces réseau.

roleArn

L'Amazon Resource Name (ARN) d'un rôle qui est autorisé à créer des interfaces réseau en votre nom.

Cet ARN doit être associé à une stratégie qui ressemble à l'exemple suivant.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/VPCDestinationENI": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface",
                    "aws:RequestTag/VPCDestinationENI": "true"
                }
            }
        }
    ]
}
```

Création d'une destination VPC en utilisant AWS CLI

L'exemple suivant montre comment créer une destination VPC à l'aide de la fonction AWS CLI.

```
aws --region regions iot create-topic-rule-destination --destination-configuration
 'vpcConfiguration={subnetIds=["subnet-
12345678910123456"],securityGroups=[],vpcId="vpc-
12345678910123456",roleArn="arn:aws:iam::123456789012:role/role-name"}'
```

Une fois que vous avez exécuté cette commande, l'état de destination VPC sera `IN_PROGRESS`. Après quelques minutes, son statut passera à `ERROR` (si la commande échoue) ou `ENABLED`. Lorsque le statut de destination est `ENABLED` défini, il est prêt à être utilisé.

Vous pouvez utiliser la commande suivante pour obtenir le statut de votre destination VPC.

```
aws --region region iot get-topic-rule-destination --arn "VPCDestinationARN"
```

Création d'une destination VPC à l'aide de la console AWS IoT Core

Les étapes suivantes décrivent comment créer une destination VPC à l'aide de la AWS IoT Core console.

1. Accédez à la AWS IoT Core console. Dans le volet de gauche, sous l'onglet Agir, sélectionnez Destinations.
2. Saisissez des valeurs pour les champs suivants.
   - ID du VPC
   - Sous-réseau IDs
   - Security Group
3. Sélectionnez un rôle qui dispose des autorisations nécessaires pour créer des interfaces réseau. L'exemple de politique précédent contient ces autorisations.

Lorsque le statut de destination du VPC est ACTIVÉ, il est prêt à être utilisé.

CloudWatch alarmes

L'action CloudWatch alarm (`cloudWatchAlarm`) modifie l'état d'une CloudWatch alarme Amazon. Vous pouvez spécifier la raison du changement d'état et la valeur dans cet appel.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`cloudwatch:SetAlarmState`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`alarmName`

Le nom de l' CloudWatch alarme.

Supporte les modèles de substitution : API et AWS CLI uniquement

`stateReason`

Raisons de la modification de l'alarme.

Prend en charge modèles de substitution : Non

`stateValue`

Valeur de l'état de l'alarme. Valeurs valides : `OK`, `ALARM`, `INSUFFICIENT_DATA`.

Prend en charge les modèles de substitution : Oui

`roleArn`

Rôle IAM qui permet d'accéder à l' CloudWatch alarme. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action d' CloudWatch alarme dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchAlarm": {
                    "alarmName": "IotAlarm",
                    "stateReason": "Temperature stabilized.",
                    "stateValue": "OK",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
```

```
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon CloudWatch ? dans le guide de CloudWatch l'utilisateur Amazon
- Utilisation des CloudWatch alarmes Amazon dans le guide de CloudWatch l'utilisateur Amazon

CloudWatch Journaux

L'action CloudWatch Logs (`cloudwatchLogs`) envoie des données à Amazon CloudWatch Logs. Vous pouvez l'utiliser `batchMode` pour télécharger et horodater plusieurs enregistrements du journal de l'appareil dans un seul message. Vous pouvez spécifier le groupe de journaux auquel l'action envoie des données.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer d'exécuter les `logs:PutLogEvents` opérations `logs:CreateLogStreamlogs:DescribeLogStreams`, et. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez une clé KMS gérée par AWS KMS key le client pour chiffrer les données du journal dans CloudWatch Logs, le service doit être autorisé à utiliser la clé KMS au nom de l'appelant. Pour plus d'informations, consultez la section Chiffrer les données des CloudWatch journaux dans les journaux à l'aide AWS KMS du guide de l'utilisateur Amazon CloudWatch Logs.

Exigences relatives au format de message MQTT pour **`batchMode`**

Si vous utilisez l'action Règle CloudWatch Logs en `batchMode` désactivant, aucune exigence de formatage des messages MQTT n'est requise. (Remarque : la valeur par défaut du paramètre `batchMode` est `false`.) Toutefois, si vous utilisez l'action Règle CloudWatch Logs en activant `batchMode` (la valeur du paramètre est`true`), les messages MQTT contenant des journaux côté appareil doivent être formatés pour contenir un horodatage et une charge utile de message.

Remarque: `timestamp` représente l'heure à laquelle l'événement s'est produit, exprimée en nombre de millisecondes après le 1er janvier 1970 00:00:00 UTC.

Voici un exemple de format de publication :

```
[
  {"timestamp": 1673520691093, "message": "Test message 1"},
  {"timestamp": 1673520692879, "message": "Test message 2"},
  {"timestamp": 1673520693442, "message": "Test message 3"}
]
```

Selon la façon dont les journaux côté appareil sont générés, il peut être nécessaire de les filtrer et de les reformater avant d'être envoyés afin de respecter cette exigence. Pour plus d'informations, consultez MQTT Message payload

Quel que soit le `batchMode` paramètre, `message` le contenu doit respecter les limites de taille des AWS IoT messages. Pour plus d'informations, consultez Points de terminaison et quotas AWS IoT Core.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`logGroupName`

Le groupe de CloudWatch journaux dans lequel l'action envoie des données.

Supporte les modèles de substitution : API et AWS CLI uniquement

`roleArn`

Rôle IAM qui autorise l'accès au groupe de CloudWatch journaux. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

(facultatif) `batchMode`

Indique si des lots d'enregistrements de journal seront extraits et chargés dans CloudWatch. Les valeurs incluent `true` ou `false` (par défaut). Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action CloudWatch Logs dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchLogs": {
                    "logGroupName": "IotLogs",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                    "batchMode": false
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon CloudWatch Logs ? dans le guide de l'utilisateur d'Amazon CloudWatch Logs

CloudWatch métriques

L'action CloudWatch metric (`cloudwatchMetric`) capture une CloudWatch métrique Amazon. Vous pouvez spécifier le namespace, le nom, la valeur, l'unité et l'horodatage de la métrique.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`cloudwatch:PutMetricData`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`metricName`

Le nom CloudWatch de la métrique.

Prend en charge les modèles de substitution : Oui

`metricNamespace`

Le nom de l'espace de noms de la CloudWatch métrique.

Prend en charge les modèles de substitution : Oui

`metricUnit`

L'unité métrique prise en charge par CloudWatch.

Prend en charge les modèles de substitution : Oui

`metricValue`

Chaîne contenant la valeur de la CloudWatch métrique.

Prend en charge les modèles de substitution : Oui

`metricTimestamp`

(Facultatif) Chaîne de caractères contenant l'horodatage, exprimé en secondes dans l'heure Unix. La valeur par défaut est l'époque Unix actuelle.

Prend en charge les modèles de substitution : Non

`roleArn`

Rôle IAM qui permet d'accéder à la CloudWatch métrique. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action CloudWatch métrique dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "IotMetric",
                    "metricNamespace": "IotNamespace",
                    "metricUnit": "Count",
                    "metricValue": "1",
                    "metricTimestamp": "1456821314",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action CloudWatch métrique avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "${topic()}",
                    "metricNamespace": "${namespace}",
                    "metricUnit": "${unit}",
                    "metricValue": "${value}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon CloudWatch ? dans le guide de CloudWatch l'utilisateur Amazon
- Utilisation des CloudWatch métriques Amazon dans le guide de CloudWatch l'utilisateur Amazon

DynamoDB

L'action DynamoDB `dynamoDB` () écrit tout ou partie d'un message MQTT dans une table Amazon DynamoDB.

Vous pouvez suivre un didacticiel qui explique comment créer et tester une règle avec une action DynamoDB. Pour de plus amples informations, veuillez consulter Tutoriel : Stockage des données de l'appareil dans une table DynamoDB.

> **Note**
>
> Cette règle écrit des données non JSON dans DynamoDB en tant que données binaires. La console DynamoDB affiche les données sous la forme de texte base64-encode.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`dynamodb:PutItem`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez une clé KMS gérée par AWS KMS key le client pour chiffrer des données inactives dans DynamoDB, le service doit être autorisé à utiliser la clé KMS au nom de l'appelant. Pour plus d'informations, consultez la section Clé KMS gérée par le client dans le guide de démarrage Amazon DynamoDB.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`tableName`

Le nom de la table DynamoDB.

Supporte les modèles de substitution : API et AWS CLI uniquement

`hashKeyField`

Nom de la clé de hachage (également appelée clé de partition).

Supporte les modèles de substitution : API et AWS CLI uniquement

`hashKeyType`

Type de données de la clé de hachage (également appelée clé de partition). Valeurs valides : `STRING`, `NUMBER`.

Supporte les modèles de substitution : API et AWS CLI uniquement

`hashKeyValue`

Valeur de la clé de hachage. Envisagez d'utiliser un modèle de substitution tel que `${topic()}` ou`${timestamp()}`.

Prend en charge les modèles de substitution : Oui

`rangeKeyField`

(Facultatif) Nom de la clé de plage (également appelée clé de tri).

Supporte les modèles de substitution : API et AWS CLI uniquement

`rangeKeyType`

(Facultatif) Type de données de la clé de plage (également appelée clé de tri). Valeurs valides : `STRING`, `NUMBER`.

Supporte les modèles de substitution : API et AWS CLI uniquement

`rangeKeyValue`

(Facultatif) Valeur de la clé de plage. Envisagez d'utiliser un modèle de substitution tel que `${topic()}` ou`${timestamp()}`.

Prend en charge les modèles de substitution : Oui

`payloadField`

(Facultatif) Nom du champ où la charge utile est écrite. Si vous omettez cette valeur, la charge utile est écrite dans la colonne nommée `payload`.

Prend en charge les modèles de substitution : Non

`operation`

(Facultatif) Type d'opération à effectuer. Valeurs valides : `INSERT`, `UPDATE`, `DELETE`.

Prend en charge les modèles de substitution : Non

`roleARN`

Rôle IAM qui autorise l'accès à la table DynamoDB. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Les données écrites dans la table DynamoDB sont le résultat de l'instruction SQL de la règle.

Exemples

L'exemple JSON suivant définit une action DynamoDB dans une règle. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDB": {
                    "tableName": "my_ddb_table",
                    "hashKeyField": "key",
                    "hashKeyValue": "${topic()}",
```

```
                    "rangeKeyField": "timestamp",
                    "rangeKeyValue": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon DynamoDB ? dans le Manuel du développeur Amazon DynamoDB
- Commencer à utiliser DynamoDB dans le manuel du développeur Amazon DynamoDB
- Tutoriel : Stockage des données de l'appareil dans une table DynamoDB

Dynamo DBv2

L'action Dynamo DBv2 (`dynamoDBv2`) écrit tout ou partie d'un message MQTT dans une table Amazon DynamoDB. Chaque attribut de la charge utile est écrit dans une colonne distincte de la base de données DynamoDB.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`dynamodb:PutItem`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- La charge du message MQTT doit contenir une clé de niveau racine qui correspond à la clé de partition primaire de la table et une clé de niveau racine qui correspond à la clé de tri primaire de la table, si elle est définie.
- Si vous utilisez une clé KMS gérée par AWS KMS key le client pour chiffrer des données inactives dans DynamoDB, le service doit être autorisé à utiliser la clé KMS au nom de l'appelant. Pour plus d'informations, consultez la section Clé KMS gérée par le client dans le guide de démarrage Amazon DynamoDB.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`putItem`

Un objet qui indique la table DynamoDB dans laquelle les données de message seront écrites. Cet objet doit contient les informations suivantes :

`tableName`

Le nom de la table DynamoDB.

Supporte les modèles de substitution : API et AWS CLI uniquement

`roleARN`

Rôle IAM qui autorise l'accès à la table DynamoDB. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Les données écrites dans la table DynamoDB sont le résultat de l'instruction SQL de la règle.

Exemples

L'exemple JSON suivant définit une DBv2 action Dynamo dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "my_ddb_table"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2",
                }
            }
```

```
        ]
    }
}
```

L'exemple JSON suivant définit une action DynamoDB avec des modèles de substitution dans une règle. AWS IoT

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2015-10-08",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "${topic()}"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon DynamoDB ? dans le Manuel du développeur Amazon DynamoDB
- Commencer à utiliser DynamoDB dans le manuel du développeur Amazon DynamoDB

Elasticsearch

L'action Elasticsearch (`elasticsearch`) écrit les données des messages MQTT dans un domaine Amazon OpenSearch Service. Vous pouvez ensuite utiliser des outils tels que OpenSearch les tableaux de bord pour interroger et visualiser les données dans OpenSearch Service.

> **Warning**
>
> L'action `Elasticsearch` ne peut être utilisée que par les actions de règle existantes. Pour créer une nouvelle action de règle ou pour mettre à jour une action de règle existante, utilisez

l'action de règle `OpenSearch` à la place. Pour de plus amples informations, veuillez consulter OpenSearch.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`es:ESHttpPut`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez une clé KMS gérée par AWS KMS key le client pour chiffrer les données au repos OpenSearch, le service doit être autorisé à utiliser la clé KMS au nom de l'appelant. Pour plus d'informations, consultez la section Chiffrement des données au repos pour Amazon OpenSearch Service dans le manuel Amazon OpenSearch Service Developer Guide.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`endpoint`

Point de terminaison de votre domaine de service.

Supporte les modèles de substitution : API et AWS CLI uniquement

`index`

Index dans lequel vous souhaitez stocker vos données.

Prend en charge les modèles de substitution : Oui

`type`

Type de document que vous stockez.

Prend en charge les modèles de substitution : Oui

`id`

Identifiant unique de chaque document.

Prend en charge les modèles de substitution : Oui

`roleARN`

Rôle IAM qui permet d'accéder au domaine de OpenSearch service. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action Elasticsearch dans une AWS IoT règle et indique comment vous pouvez spécifier les champs de cette action. `elasticsearch` Pour de plus amples informations, veuillez consulter ElasticsearchAction.

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "my-type",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action Elasticsearch avec des modèles de substitution dans une AWS IoT règle.

```
{
```

```
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

Consultez aussi

- OpenSearch
- Qu'est-ce qu'Amazon OpenSearch Service ?

HTTP

L'action HTTPS (`http`) envoie les données d'un message MQTT à une application ou un service Web.

Prérequis

Ce rôle d’action possède les critères suivants :

- Vous devez confirmer et activer les points de terminaison HTTPS avant que le moteur de règles puisse les utiliser. Pour de plus amples informations, veuillez consulter Utilisation des destinations des règles de rubrique HTTP.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`url`

Point de terminaison HTTPS à l'endroit où le message est envoyé à l'aide de la méthode HTTP POST. Si vous utilisez une adresse IP à la place d'un nom d'hôte, il doit s'agir d'une IPv4 adresse. IPv6 les adresses ne sont pas prises en charge.

Prend en charge les modèles de substitution : Oui

`confirmationUrl`

(Facultatif) Si cela est spécifié, AWS IoT utilise l'URL de confirmation pour créer une destination de règle de sujet correspondante. Vous devez activer la destination de la règle avant de l'utiliser dans une action HTTP. Pour de plus amples informations, veuillez consulter Utilisation des destinations des règles de rubrique HTTP. Si vous utilisez des modèles de substitution, vous devez créer manuellement des destinations de règles de rubrique avant que l'action `http` puisse être utilisée. `confirmationUrl` doit être un préfixe de `url`.

La relation entre `url` et `confirmationUrl` est décrite par les éléments suivants :

- S'il `url` est codé en dur et n'`confirmationUrl`est pas fourni, nous traitons implicitement le `url` champ comme le. `confirmationUrl` AWS IoT crée une destination de règle de sujet pour`url`.
- Si `url` et `confirmationUrl` sont codés en dur, cela `url` doit commencer `confirmationUrl` par. AWS IoT crée une destination de règle de sujet pour`confirmationUrl`.
- Si `url` contient un modèle de substitution, vous devez spécifier `confirmationUrl` et `url` doit commencer par `confirmationUrl`. Si `confirmationUrl` contient des modèles de substitution, vous devez créer manuellement des destinations de règle de rubrique avant que l'action `http` puisse être utilisée. S'il `confirmationUrl` ne contient pas de modèles de substitution, AWS IoT crée une destination de règle de sujet pour`confirmationUrl`.

Prend en charge les modèles de substitution : Oui

`headers`

(Facultatif) La liste des en-têtes à inclure dans les requêtes HTTP adressées au point de terminaison. Chaque entête doit contenir les informations suivantes.

`key`

La clé de l'en-tête.

Prend en charge les modèles de substitution: Non

`value`

Valeur de l'en-tête.

Prend en charge les modèles de substitution: Oui

> **Note**
>
> Le type de contenu par défaut est application/json lorsque la charge utile est au format JSON. Sinon, il s'agit de application/octet-stream. Vous pouvez le remplacer en spécifiant le type de contenu exact dans l'en-tête avec le type de contenu clé (insensible à la casse).

`auth`

(Facultatif) L'authentification utilisée par le moteur de règles pour se connecter à l'URL du point final spécifié dans `url` l'argument. Actuellement, Signature Version 4 est le seul type d'authentification pris en charge. Pour de plus amples informations, veuillez consulter Autorisation HTTP.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une AWS IoT règle avec une action HTTP.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
```

```
                        "value": "static_header_value"
                    },
                    {
                        "key": "substitutable_header_key",
                        "value": "${value_from_payload}"
                    }
                ]
            }
        }
    ]
  }
}
```

Logique de nouvelle tentative d'action HTTP

Le moteur de AWS IoT règles tente à nouveau l'action HTTP conformément aux règles suivantes :

- Le moteur de règles essaie d'envoyer un message au moins une fois.
- Le moteur de règles effectue au plus deux nouvelles tentatives. Le nombre maximum de nouvelles tentatives est trois.
- Le moteur de règles n'effectue pas de nouvelle tentative si :
  - L'essai précédent a fourni une réponse de plus de 16 384 octets.
  - Le service web ou l'application en aval ferme la connexion TCP après la tentative.
  - Le temps total d'exécution d'une demande avec tentatives a dépassé la limite de temporisation de la demande.
  - La requête renvoie un code d'état HTTP autre que 429, 500-599.

> **Note**
>
> Les coûts standard de transfert de données s'appliquent aux nouvelles tentatives.

Consultez aussi

- Utilisation des destinations des règles de rubrique HTTP
- Acheminez les données directement depuis AWS IoT Core vos services Web dans l'Internet des objets sur le AWS blog

Utilisation des destinations des règles de rubrique HTTP

Une destination de règle de sujet HTTP est un service Web vers lequel le moteur de règles peut acheminer les données d'une règle de sujet. Une AWS IoT Core ressource décrit le service Web pour AWS IoT. Les ressources de destination des règles thématiques peuvent être partagées selon différentes règles.

Avant de AWS IoT Core pouvoir envoyer des données à un autre service Web, celui-ci doit confirmer qu'il peut accéder au point de terminaison du service.

Dans ce chapitre :

- Vue d'ensemble des destinations de règle de rubrique HTTP
- Gestion des destinations des règles de rubrique HTTP
- Autorités de certification prises en charge par les points de terminaison HTTPS dans les destinations des règles thématiques

Vue d'ensemble des destinations de règle de rubrique HTTP

Une destination de règle de rubrique HTTP fait référence à un service Web qui prend en charge une URL de confirmation et une ou plusieurs collectes de données URLs. La ressource de destination des règles de rubrique HTTP contient l'URL de confirmation de votre service Web. Lorsque vous configurez une action de règle de rubrique HTTP, vous spécifiez l'URL réelle du point de terminaison qui doit recevoir les données ainsi que l'URL de confirmation du service Web. Une fois votre destination confirmée, la règle du sujet envoie le résultat de l'instruction SQL au point de terminaison HTTPS (et non à l'URL de confirmation).

La destination d'une règle de thème HTTP peut se trouver dans l'un des états suivants :

ENABLED

La destination a été confirmée et peut être utilisée par une action de règle. L'état d'une destination doit être `ENABLED` (ACTIVÉ) pour qu'elle soit utilisée dans une règle. Vous pouvez uniquement activer une destination dont l'état est DISABLED (DÉSACTIVÉ).

DISABLED

La destination a été confirmée mais elle ne peut pas être utilisée par une action de règle. Cet état est utile si vous souhaitez empêcher temporairement le trafic vers votre point de terminaison sans avoir à passer à nouveau par le processus de confirmation. Vous pouvez uniquement désactiver une destination dont l'état est ENABLED (ACTIVÉ).

EN_COURS

La confirmation de la destination est en cours.

ERROR

La confirmation de la destination a expiré.

Une fois que la destination d'une règle de thème HTTP a été confirmée et activée, elle peut être utilisée avec n'importe quelle règle de votre compte.

Les sections suivantes décrivent les actions les plus courantes sur les destinations des règles de rubrique HTTP.

Gestion des destinations des règles de rubrique HTTP

Vous pouvez utiliser les opérations suivantes pour gérer les destinations de vos règles de rubrique HTTP.

Dans cette rubrique :

- Création de destinations de règles de rubrique HTTP
- Confirmation des destinations des règles de rubrique HTTP
- Envoi d'une nouvelle demande de confirmation
- Désactivation et suppression de la destination d'une règle thématique

Création de destinations de règles de rubrique HTTP

Vous créez une destination de règle de sujet HTTP en appelant l'`CreateTopicRuleDestination`opération ou en utilisant la AWS IoT console.

Après avoir créé une destination, AWS IoT envoie une demande de confirmation à l'URL de confirmation. Le format de la demande de confirmation est le suivant :

```
HTTP POST {confirmationUrl}/?confirmationToken={confirmationToken}
Headers:
x-amz-rules-engine-message-type: DestinationConfirmation
x-amz-rules-engine-destination-arn:"arn:aws:iot:us-east-1:123456789012:ruledestination/
http/7a280e37-b9c6-47a2-a751-0703693f46e4"
Content-Type: application/json
Body:
```

```
{
    "arn":"arn:aws:iot:us-east-1:123456789012:ruledestination/http/7a280e37-b9c6-47a2-
a751-0703693f46e4",
    "confirmationToken": "AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "enableUrl": "https://iot.us-east-1.amazonaws.com/confirmdestination/
AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "messageType": "DestinationConfirmation"
}
```

Le contenu de la demande de confirmation comprend les informations suivantes :

arn

L'Amazon Resource Name (ARN) de la destination de la règle de rubrique à confirmer.

confirmationToken

Le jeton de confirmation envoyé par AWS IoT Core. Dans l'exemple, le jeton est tronqué. Votre jeton sera plus long. Vous aurez besoin de ce jeton pour confirmer votre destination AWS IoT Core.

enableUrl

L'URL à laquelle vous accédez pour confirmer la destination d'une règle de rubrique.

messageType

Type du message.

Confirmation des destinations des règles de rubrique HTTP

Pour terminer le processus de confirmation du point de terminaison, si vous utilisez le AWS CLI, vous devez effectuer les étapes suivantes une fois que votre URL de confirmation a reçu la demande de confirmation.

1. Confirmez que la destination accepte de recevoir des messages

   Pour confirmer que la destination de la règle thématique est prête à recevoir des messages IoT, appelez le `enableUrl` dans la demande de confirmation ou effectuez l'opération `ConfirmTopicRuleDestination` API et transmettez le contenu `confirmationToken` de la demande de confirmation.
2. Définir le statut des règles du sujet sur Activé

Après avoir confirmé que la destination peut recevoir des messages, vous devez exécuter l'opération d'`UpdateTopicRuleDestination`API pour définir le statut de la règle du sujet sur`ENABLED`.

Si vous utilisez la AWS IoT console, copiez-la `confirmationToken` et collez-la dans la boîte de dialogue de confirmation de la destination dans la AWS IoT console. Vous pouvez ensuite activer la règle du sujet.

Envoi d'une nouvelle demande de confirmation

Pour activer un nouveau message de confirmation pour une destination, appelez `UpdateTopicRuleDestination` et réglez le statut de la règle de thème destination sur `IN_PROGRESS`.

Répétez le processus de confirmation après avoir envoyé une nouvelle demande de confirmation.

Désactivation et suppression de la destination d'une règle thématique

Pour désactiver une destination, appelez `UpdateTopicRuleDestination` et définissez l'état de la destination de règle de rubrique sur `DISABLED`. Une règle de rubrique à l'état DÉSACTIVÉ peut être réactivée sans qu'il soit nécessaire d'envoyer une nouvelle demande de confirmation.

Pour supprimer une destination de règle de rubrique, appelez `DeleteTopicRuleDestination`.

Autorités de certification prises en charge par les points de terminaison HTTPS dans les destinations des règles thématiques

Les autorités de certification suivantes sont prises en charge par les points de terminaison HTTPS dans les destinations des règles thématiques. Vous pouvez choisir l'une de ces autorités de certification prises en charge. Les signatures sont fournies à titre de référence. Notez que vous ne pouvez pas utiliser de certificats auto-signés car ils ne fonctionneront pas.

Aidez-nous à améliorer ce sujet

Dites-nous ce que vous en pensez

```
Alias name: swisssignplatinumg2ca
Certificate fingerprints:
  MD5:  C9:98:27:77:28:1E:3D:0E:15:3C:84:00:B8:85:03:E6
```

```
  SHA1: 56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
  SHA256:
 3B:22:2E:56:67:11:E9:92:30:0D:C0:B1:5A:B9:47:3D:AF:DE:F8:C8:4D:0C:EF:7D:33:17:B4:C1:82:1D:14:3

Alias name: hellenicacademicandresearchinstitutionsrootca2011
Certificate fingerprints:
  MD5:  73:9F:4C:4B:73:5B:79:E9:FA:BA:1C:EF:6E:CB:D5:C9
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7

Alias name: teliasonerarootcav1
Certificate fingerprints:
  MD5:  37:41:49:1B:18:56:9A:26:F5:AD:C2:66:FB:40:A5:4C
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8

Alias name: geotrustprimarycertificationauthority
Certificate fingerprints:
  MD5:  02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6

Alias name: trustisfpsrootca
Certificate fingerprints:
  MD5:  30:C9:E7:1E:6B:E6:14:EB:65:B2:16:69:20:31:67:4D
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7

Alias name: quovadisrootca3g3
Certificate fingerprints:
  MD5:  DF:7D:B9:AD:54:6F:68:A1:DF:89:57:03:97:43:B0:D7
  SHA1: 48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
  SHA256:
 88:EF:81:DE:20:2E:B0:18:45:2E:43:F8:64:72:5C:EA:5F:BD:1F:C2:D9:D2:05:73:07:09:C5:D8:B8:69:0F:4

Alias name: buypassclass2ca
Certificate fingerprints:
  MD5:  46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
```

```
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4

Alias name: secureglobalca
Certificate fingerprints:
  MD5:  CF:F4:27:0D:D4:ED:DC:65:16:49:6D:3D:DA:BF:6E:DE
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6

Alias name: chunghwaepkirootca
Certificate fingerprints:
  MD5:  1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D

Alias name: verisignclass2g2ca
Certificate fingerprints:
  MD5:  2D:BB:E5:25:D3:D1:65:82:3A:B7:0E:FA:E6:EB:E2:E1
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A

Alias name: szafirrootca2
Certificate fingerprints:
  MD5:  11:64:C1:89:B0:24:B1:8C:B1:07:7E:89:9E:51:9E:99
  SHA1: E2:52:FA:95:3F:ED:DB:24:60:BD:6E:28:F3:9C:CC:CF:5E:B3:3F:DE
  SHA256:
 A1:33:9D:33:28:1A:0B:56:E5:57:D3:D3:2B:1C:E7:F9:36:7E:B0:94:BD:5F:A7:2A:7E:50:04:C8:DE:D7:CA:F

Alias name: quovadisrootca1g3
Certificate fingerprints:
  MD5:  A4:BC:5B:3F:FE:37:9A:FA:64:F0:E2:FA:05:3D:0B:AB
  SHA1: 1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
  SHA256:
 8A:86:6F:D1:B2:76:B5:7E:57:8E:92:1C:65:82:8A:2B:ED:58:E9:F2:F2:88:05:41:34:B7:F1:F4:BF:C9:CC:7

Alias name: utndatacorpsgcca
Certificate fingerprints:
  MD5:  B3:A5:3E:77:21:6D:AC:4A:C0:C9:FB:D5:41:3D:CA:06
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
```

```
Alias name: autoridaddecertificacionfirmaprofesionalcifa62634068
Certificate fingerprints:
  MD5:  73:3A:74:7A:EC:BB:A3:96:A6:C2:E4:E2:C8:9B:C0:C3
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E

Alias name: securesignrootca11
Certificate fingerprints:
  MD5:  B7:52:74:E2:92:B4:80:93:F2:75:E4:CC:D7:F2:EA:26
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1

Alias name: amazon-ca-g4-acm2
Certificate fingerprints:
  MD5:  B2:F1:03:2B:93:64:05:80:B8:A8:17:36:B9:1B:52:3C
  SHA1: A7:E6:45:32:1F:7A:B7:AD:C0:70:EA:73:5F:AB:ED:C3:DA:B4:D0:C8
  SHA256:
 D7:A8:7C:69:95:D0:E2:04:2A:32:70:A7:E2:87:FE:A7:E8:F4:C1:70:62:F7:90:C3:EB:BB:53:F2:AC:39:26:B

Alias name: isrgrootx1
Certificate fingerprints:
  MD5:  0C:D2:F9:E0:DA:17:73:E9:ED:86:4D:A5:E3:70:E7:4E
  SHA1: CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
  SHA256:
 96:BC:EC:06:26:49:76:F3:74:60:77:9A:CF:28:C5:A7:CF:E8:A3:C0:AA:E1:1A:8F:FC:EE:05:C0:BD:DF:08:C

Alias name: amazon-ca-g4-acm1
Certificate fingerprints:
  MD5:  E2:F1:18:19:61:5C:43:E0:D4:A8:5D:0B:FA:7C:89:1B
  SHA1: F2:0D:28:B6:29:C2:2C:5E:84:05:E6:02:4D:97:FE:8F:A0:84:93:A0
  SHA256:
 B0:11:A4:F7:29:6C:74:D8:2B:F5:62:DF:87:D7:28:C7:1F:B5:8C:F4:E6:73:F2:78:FC:DA:F3:FF:83:A6:8C:8

Alias name: etugracertificationauthority
Certificate fingerprints:
  MD5:  B8:A1:03:63:B0:BD:21:71:70:8A:6F:13:3A:BB:79:49
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3

Alias name: geotrustuniversalca2
```

```
Certificate fingerprints:
  MD5:  34:FC:B8:D0:36:DB:9E:14:B3:C2:F2:DB:8F:E4:94:C7
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0

Alias name: digicertglobalrootca
Certificate fingerprints:
  MD5:  79:E4:A9:84:0D:7D:3A:96:D7:C0:4F:E2:43:4C:89:2E
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6

Alias name: staatdernederlandenevrootca
Certificate fingerprints:
  MD5:  FC:06:AF:7B:E8:1A:F1:9A:B4:E8:D2:70:1F:C0:F5:BA
  SHA1: 76:E2:7E:C1:4F:DB:82:C1:C0:A6:75:B5:05:BE:3D:29:B4:ED:DB:BB
  SHA256:
 4D:24:91:41:4C:FE:95:67:46:EC:4C:EF:A6:CF:6F:72:E2:8A:13:29:43:2F:9D:8A:90:7A:C4:CB:5D:AD:C1:5

Alias name: utnuserfirstclientauthemailca
Certificate fingerprints:
  MD5:  D7:34:3D:EF:1D:27:09:28:E1:31:02:5B:13:2B:DD:F7
  SHA1: B1:72:B1:A5:6D:95:F9:1F:E5:02:87:E1:4D:37:EA:6A:44:63:76:8A
  SHA256:
 43:F2:57:41:2D:44:0D:62:74:76:97:4F:87:7D:A8:F1:FC:24:44:56:5A:36:7A:E6:0E:DD:C2:7A:41:25:31:A

Alias name: actalisauthenticationrootca
Certificate fingerprints:
  MD5:  69:C1:0D:4F:07:A3:1B:C3:FE:56:3D:04:BC:11:F6:A6
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6

Alias name: amazonrootca4
Certificate fingerprints:
  MD5:  89:BC:27:D5:EB:17:8D:06:6A:69:D5:FD:89:47:B4:CD
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9

Alias name: amazonrootca3
Certificate fingerprints:
  MD5:  A0:D4:EF:0B:F7:B5:D8:49:95:2A:EC:F5:C4:FC:81:87
```

```
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A

Alias name: amazonrootca2
Certificate fingerprints:
  MD5:  C8:E5:8D:CE:A8:42:E2:7A:C0:2A:5C:7C:9E:26:BF:66
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B

Alias name: amazonrootca1
Certificate fingerprints:
  MD5:  43:C6:BF:AE:EC:FE:AD:2F:18:C6:88:68:30:FC:C8:E6
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6

Alias name: affirmtrustpremium
Certificate fingerprints:
  MD5:  C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9

Alias name: keynectisrootca
Certificate fingerprints:
  MD5:  CC:4D:AE:FB:30:6B:D8:38:FE:50:EB:86:61:4B:D2:26
  SHA1: 9C:61:5C:4D:4D:85:10:3A:53:26:C2:4D:BA:EA:E4:A2:D2:D5:CC:97
  SHA256:
 42:10:F1:99:49:9A:9A:C3:3C:8D:E0:2B:A6:DB:AA:14:40:8B:DD:8A:6E:32:46:89:C1:92:2D:06:97:15:A3:3

Alias name: equifaxsecureglobalebusinessca1
Certificate fingerprints:
  MD5:  51:F0:2A:33:F1:F5:55:39:07:F2:16:7A:47:C7:5D:63
  SHA1: 3A:74:CB:7A:47:DB:70:DE:89:1F:24:35:98:64:B8:2D:82:BD:1A:36
  SHA256:
 86:AB:5A:65:71:D3:32:9A:BC:D2:E4:E6:37:66:8B:A8:9C:73:1E:C2:93:B6:CB:A6:0F:71:63:40:A0:91:CE:A

Alias name: affirmtrustpremiumca
Certificate fingerprints:
  MD5:  C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
```

```
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9

Alias name: baltimorecodesigningca
Certificate fingerprints:
  MD5:  90:F5:28:49:56:D1:5D:2C:B0:53:D4:4B:EF:6F:90:22
  SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
  SHA256:
 A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:8

Alias name: gdcatrustauthr5root
Certificate fingerprints:
  MD5:  63:CC:D9:3D:34:35:5C:6F:53:A3:E2:08:70:48:1F:B4
  SHA1: 0F:36:38:5B:81:1A:25:C3:9B:31:4E:83:CA:E9:34:66:70:CC:74:B4
  SHA256:
 BF:FF:8F:D0:44:33:48:7D:6A:8A:A6:0C:1A:29:76:7A:9F:C2:BB:B0:5E:42:0F:71:3A:13:B9:92:89:1D:38:9

Alias name: certinomisrootca
Certificate fingerprints:
  MD5:  14:0A:FD:8D:A8:28:B5:38:69:DB:56:7E:61:22:03:3F
  SHA1: 9D:70:BB:01:A5:A4:A0:18:11:2E:F7:1C:01:B9:32:C5:34:E7:88:A8
  SHA256:
 2A:99:F5:BC:11:74:B7:3C:BB:1D:62:08:84:E0:1C:34:E5:1C:CB:39:78:DA:12:5F:0E:33:26:88:83:BF:41:5

Alias name: verisignclass3publicprimarycertificationauthorityg5
Certificate fingerprints:
  MD5:  CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D

Alias name: verisignclass3publicprimarycertificationauthorityg4
Certificate fingerprints:
  MD5:  3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7

Alias name: verisignclass3publicprimarycertificationauthorityg3
Certificate fingerprints:
  MD5:  CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
```

```
Alias name: swisssignsilverg2ca
Certificate fingerprints:
  MD5:  E0:06:A1:C9:7D:CF:C9:FC:0D:C0:56:75:96:D8:62:13
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D

Alias name: swisssignsilvercag2
Certificate fingerprints:
  MD5:  E0:06:A1:C9:7D:CF:C9:FC:0D:C0:56:75:96:D8:62:13
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D

Alias name: atostrustedroot2011
Certificate fingerprints:
  MD5:  AE:B9:C4:32:4B:AC:7F:5D:66:CC:77:94:BB:2A:77:56
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7

Alias name: comodoecccertificationauthority
Certificate fingerprints:
  MD5:  7C:62:FF:74:9D:31:53:5E:68:4A:D5:78:AA:1E:BF:23
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C

Alias name: securetrustca
Certificate fingerprints:
  MD5:  DC:32:C3:A7:6D:25:57:C7:68:09:9D:EA:2D:A9:A2:D1
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7

Alias name: soneraclass1ca
Certificate fingerprints:
  MD5:  33:B7:84:F5:5F:27:D7:68:27:DE:14:DE:12:2A:ED:6F
  SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
  SHA256:
 CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3

Alias name: cadisigrootr2
```

```
Certificate fingerprints:
  MD5:  26:01:FB:D8:27:A7:17:9A:45:54:38:1A:43:01:3B:03
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0

Alias name: cadisigrootr1
Certificate fingerprints:
  MD5:  BE:EC:11:93:9A:F5:69:21:BC:D7:C1:C0:67:89:CC:2A
  SHA1: 8E:1C:74:F8:A6:20:B9:E5:8A:F4:61:FA:EC:2B:47:56:51:1A:52:C6
  SHA256:
 F9:6F:23:F4:C3:E7:9C:07:7A:46:98:8D:5A:F5:90:06:76:A0:F0:39:CB:64:5D:D1:75:49:B2:16:C8:24:40:C

Alias name: verisignclass3g5ca
Certificate fingerprints:
  MD5:  CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D

Alias name: utnuserfirsthardwareca
Certificate fingerprints:
  MD5:  4C:56:41:E5:0D:BB:2B:E8:CA:A3:ED:18:08:AD:43:39
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3

Alias name: addtrustqualifiedca
Certificate fingerprints:
  MD5:  27:EC:39:47:CD:DA:5A:AF:E2:9A:01:65:21:A9:4C:BB
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1

Alias name: verisignclass3g3ca
Certificate fingerprints:
  MD5:  CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4

Alias name: thawtepersonalfreemailca
Certificate fingerprints:
  MD5:  53:4B:1D:17:58:58:1A:30:A1:90:F8:6E:5C:F2:CF:65
```

```
  SHA1: E6:18:83:AE:84:CA:C1:C1:CD:52:AD:E8:E9:25:2B:45:A6:4F:B7:E2
  SHA256:
 5B:38:BD:12:9E:83:D5:A0:CA:D2:39:21:08:94:90:D5:0D:4A:AE:37:04:28:F8:DD:FF:FF:FA:4C:15:64:E1:8

Alias name: certplusclass3pprimaryca
Certificate fingerprints:
  MD5:  E1:4B:52:73:D7:1B:DB:93:30:E5:BD:E4:09:6E:BE:FB
  SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
  SHA256:
 CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:8

Alias name: swisssigngoldg2ca
Certificate fingerprints:
  MD5:  24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9

Alias name: swisssigngoldcag2
Certificate fingerprints:
  MD5:  24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9

Alias name: dtrustrootclass3ca22009
Certificate fingerprints:
  MD5:  CD:E0:25:69:8D:47:AC:9C:89:35:90:F7:FD:51:3D:2F
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C

Alias name: acraizfnmtrcm
Certificate fingerprints:
  MD5:  E2:09:04:B4:D3:BD:D1:A0:14:FD:1A:D2:47:C4:57:1D
  SHA1: EC:50:35:07:B2:15:C4:95:62:19:E2:A8:9A:5B:42:99:2C:4C:2C:20
  SHA256:
 EB:C5:57:0C:29:01:8C:4D:67:B1:AA:12:7B:AF:12:F7:03:B4:61:1E:BC:17:B7:DA:B5:57:38:94:17:9B:93:F

Alias name: securitycommunicationevrootca1
Certificate fingerprints:
  MD5:  22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
```

```
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3

Alias name: starfieldclass2ca
Certificate fingerprints:
  MD5:  32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5

Alias name: opentrustrootcag3
Certificate fingerprints:
  MD5:  21:37:B4:17:16:92:7B:67:46:70:A9:96:D7:A8:13:24
  SHA1: 6E:26:64:F3:56:BF:34:55:BF:D1:93:3F:7C:01:DE:D8:13:DA:8A:A6
  SHA256:
 B7:C3:62:31:70:6E:81:07:8C:36:7C:B8:96:19:8F:1E:32:08:DD:92:69:49:DD:8F:57:09:A4:10:F7:5B:62:9

Alias name: opentrustrootcag2
Certificate fingerprints:
  MD5:  57:24:B6:59:24:6B:AE:C8:FE:1C:0C:20:F2:C0:4E:EB
  SHA1: 79:5F:88:60:C5:AB:7C:3D:92:E6:CB:F4:8D:E1:45:CD:11:EF:60:0B
  SHA256:
 27:99:58:29:FE:6A:75:15:C1:BF:E8:48:F9:C4:76:1D:B1:6C:22:59:29:25:7B:F4:0D:08:94:F2:9E:A8:BA:F

Alias name: buypassclass2rootca
Certificate fingerprints:
  MD5:  46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4

Alias name: opentrustrootcag1
Certificate fingerprints:
  MD5:  76:00:CC:81:29:CD:55:5E:88:6A:7A:2E:F7:4D:39:DA
  SHA1: 79:91:E8:34:F7:E2:EE:DD:08:95:01:52:E9:55:2D:14:E9:58:D5:7E
  SHA256:
 56:C7:71:28:D9:8C:18:D9:1B:4C:FD:FF:BC:25:EE:91:03:D4:75:8E:A2:AB:AD:82:6A:90:F3:45:7D:46:0E:B

Alias name: globalsignr2ca
Certificate fingerprints:
  MD5:  94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
```

```
Alias name: buypassclass3rootca
Certificate fingerprints:
  MD5:  3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4

Alias name: ecacc
Certificate fingerprints:
  MD5:  EB:F5:9D:29:0D:61:F9:42:1F:7C:C2:BA:6D:E3:15:09
  SHA1: 28:90:3A:63:5B:52:80:FA:E6:77:4C:0B:6D:A7:D6:BA:A6:4A:F2:E8
  SHA256:
 88:49:7F:01:60:2F:31:54:24:6A:E2:8C:4D:5A:EF:10:F1:D8:7E:BB:76:62:6F:4A:E0:B7:F9:5B:A7:96:87:9

Alias name: epkirootcertificationauthority
Certificate fingerprints:
  MD5:  1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D

Alias name: verisignclass1g2ca
Certificate fingerprints:
  MD5:  DB:23:3D:F9:69:FA:4B:B9:95:80:44:73:5E:7D:41:83
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7

Alias name: certigna
Certificate fingerprints:
  MD5:  AB:57:A6:5B:7D:42:82:19:B5:D8:58:26:28:5E:FD:FF
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2

Alias name: camerfirmaglobalchambersignroot
Certificate fingerprints:
  MD5:  C5:E6:7B:BF:06:D0:4F:43:ED:C4:7A:65:8A:FB:6B:19
  SHA1: 33:9B:6B:14:50:24:9B:55:7A:01:87:72:84:D9:E0:2F:C3:D2:D8:E9
  SHA256:
 EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:39:DE:1E:A5:FE:64:91:41:D1:02:8B:7D:11:C0:B2:29:8C:E

Alias name: cfcaevroot
```

```
Certificate fingerprints:
  MD5:  74:E1:B6:ED:26:7A:7A:44:30:33:94:AB:7B:27:81:30
  SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
  SHA256:
 5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:F

Alias name: soneraclass2rootca
Certificate fingerprints:
  MD5:  A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2

Alias name: certumtrustednetworkca
Certificate fingerprints:
  MD5:  D5:E9:81:40:C5:18:69:FC:46:2C:89:75:62:0F:AA:78
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8

Alias name: securitycommunicationrootca2
Certificate fingerprints:
  MD5:  6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F

Alias name: globalsigneccrootcar5
Certificate fingerprints:
  MD5:  9F:AD:3B:1C:02:1E:8A:BA:17:74:38:81:0C:A2:BC:08
  SHA1: 1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
  SHA256:
 17:9F:BC:14:8A:3D:D0:0F:D2:4E:A1:34:58:CC:43:BF:A7:F5:9C:81:82:D7:83:A5:13:F6:EB:EC:10:0C:89:2

Alias name: globalsigneccrootcar4
Certificate fingerprints:
  MD5:  20:F0:27:68:D1:7E:A0:9D:0E:E6:2A:CA:DF:5C:89:8E
  SHA1: 69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
  SHA256:
 BE:C9:49:11:C2:95:56:76:DB:6C:0A:55:09:86:D7:6E:3B:A0:05:66:7C:44:2C:97:62:B4:FB:B7:73:DE:22:8

Alias name: chambersofcommerceroot2008
Certificate fingerprints:
  MD5:  5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
```

```
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C

Alias name: pscprocert
Certificate fingerprints:
  MD5:  E6:24:E9:12:01:AE:0C:DE:8E:85:C4:CE:A3:12:DD:EC
  SHA1: 70:C1:8D:74:B4:28:81:0A:E4:FD:A5:75:D7:01:9F:99:B0:3D:50:74
  SHA256:
 3C:FC:3C:14:D1:F6:84:FF:17:E3:8C:43:CA:44:0C:00:B9:67:EC:93:3E:8B:FE:06:4C:A1:D7:2C:90:F2:AD:B

Alias name: thawteprimaryrootcag3
Certificate fingerprints:
  MD5:  FB:1B:5D:43:8A:94:CD:44:C6:76:F2:43:4B:47:E7:31
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4

Alias name: quovadisrootca
Certificate fingerprints:
  MD5:  27:DE:36:FE:72:B7:00:03:00:9D:F4:F0:1E:6C:04:24
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7

Alias name: thawteprimaryrootcag2
Certificate fingerprints:
  MD5:  74:9D:EA:60:24:C4:FD:22:53:3E:CC:3A:72:D9:29:4F
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5

Alias name: deprecateditsecca
Certificate fingerprints:
  MD5:  A5:96:0C:F6:B5:AB:27:E5:01:C6:00:88:9E:60:33:E5
  SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
  SHA256:
 9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:C

Alias name: usertrustrsacertificationauthority
Certificate fingerprints:
  MD5:  1B:FE:69:D1:91:B7:19:33:A3:72:A8:0F:E1:55:E5:B5
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
```

```
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D

Alias name: entrustrootcag2
Certificate fingerprints:
  MD5:  4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3

Alias name: networksolutionscertificateauthority
Certificate fingerprints:
  MD5:  D3:F3:A6:16:C0:FA:6B:1D:59:B1:2D:96:4D:0E:11:2E
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0

Alias name: trustcenterclass4caii
Certificate fingerprints:
  MD5:  9D:FB:F9:AC:ED:89:33:22:F4:28:48:83:25:23:5B:E0
  SHA1: A6:9A:91:FD:05:7F:13:6A:42:63:0B:B1:76:0D:2D:51:12:0C:16:50
  SHA256:
 32:66:96:7E:59:CD:68:00:8D:9D:D3:20:81:11:85:C7:04:20:5E:8D:95:FD:D8:4F:1C:7B:31:1E:67:04:FC:3

Alias name: oistewisekeyglobalrootgaca
Certificate fingerprints:
  MD5:  BC:6C:51:33:A7:E9:D3:66:63:54:15:72:1B:21:92:93
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F

Alias name: verisignuniversalrootcertificationauthority
Certificate fingerprints:
  MD5:  8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3

Alias name: ttelesecglobalrootclass3ca
Certificate fingerprints:
  MD5:  CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
```

```
Alias name: starfieldservicesrootg2ca
Certificate fingerprints:
  MD5:  17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B

Alias name: addtrustexternalroot
Certificate fingerprints:
  MD5:  1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F

Alias name: turktrustelektroniksertifikahizmetsaglayicisih5
Certificate fingerprints:
  MD5:  DA:70:8E:F0:22:DF:93:26:F6:5F:9F:D3:15:06:52:4E
  SHA1: C4:18:F6:4D:46:D1:DF:00:3D:27:30:13:72:43:A9:12:11:C6:75:FB
  SHA256:
 49:35:1B:90:34:44:C1:85:CC:DC:5C:69:3D:24:D8:55:5C:B2:08:D6:A8:14:13:07:69:9F:4A:F0:63:19:9D:7

Alias name: camerfirmachambersca
Certificate fingerprints:
  MD5:  5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C

Alias name: certsignrootca
Certificate fingerprints:
  MD5:  18:98:C0:D6:E9:3A:FC:F9:B0:F5:0C:F7:4B:01:44:17
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B

Alias name: verisignuniversalrootca
Certificate fingerprints:
  MD5:  8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3

Alias name: geotrustuniversalca
```

```
Certificate fingerprints:
  MD5:  92:65:58:8B:A2:1A:31:72:73:68:5C:B4:A5:7A:07:48
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1



Alias name: luxtrustglobalroot2
Certificate fingerprints:
  MD5:  B2:E1:09:00:61:AF:F7:F1:91:6F:C4:AD:8D:5E:3B:7C
  SHA1: 1E:0E:56:19:0A:D1:8B:25:98:B2:04:44:FF:66:8A:04:17:99:5F:3F
  SHA256:
 54:45:5F:71:29:C2:0B:14:47:C4:18:F9:97:16:8F:24:C5:8F:C5:02:3B:F5:DA:5B:E2:EB:6E:1D:D8:90:2E:D



Alias name: twcaglobalrootca
Certificate fingerprints:
  MD5:  F9:03:7E:CF:E6:9E:3C:73:7A:2A:90:07:69:FF:2B:96
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1

Alias name: tubitakkamusmsslkoksertifikasisurum1
Certificate fingerprints:
  MD5:  DC:00:81:DC:69:2F:3E:2F:B0:3B:F6:3D:5A:91:8E:49
  SHA1: 31:43:64:9B:EC:CE:27:EC:ED:3A:3F:0B:8F:0D:E4:E8:91:DD:EE:CA
  SHA256:
 46:ED:C3:68:90:46:D5:3A:45:3F:B3:10:4A:B8:0D:CA:EC:65:8B:26:60:EA:16:29:DD:7E:86:79:90:64:87:1

Alias name: affirmtrustnetworkingca
Certificate fingerprints:
  MD5:  42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1

Alias name: affirmtrustcommercialca
Certificate fingerprints:
  MD5:  82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6:B7
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
```

```
Alias name: godaddyrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D

Alias name: starfieldrootg2ca
Certificate fingerprints:
  MD5:  D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F

Alias name: dtrustrootclass3ca2ev2009
Certificate fingerprints:
  MD5:  AA:C6:43:2C:5E:2D:CD:C4:34:C0:50:4F:11:02:4F:B6
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8

Alias name: buypassclass3ca
Certificate fingerprints:
  MD5:  3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4

Alias name: verisignclass2g3ca
Certificate fingerprints:
  MD5:  F8:BE:C4:63:22:C9:A8:46:74:8B:B8:1D:1E:4A:2B:F6
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B

Alias name: digicerttrustedrootg4
Certificate fingerprints:
  MD5:  78:F2:FC:AA:60:1F:2F:B4:EB:C9:37:BA:53:2E:75:49
  SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
  SHA256:
 55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:8

Alias name: quovadisrootca2g3
```

```
Certificate fingerprints:
  MD5:  AF:0C:86:6E:BF:40:2D:7F:0B:3E:12:50:BA:12:3D:06
  SHA1: 09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
  SHA256:
 8F:E4:FB:0A:F9:3A:4D:0D:67:DB:0B:EB:B2:3E:37:C7:1B:F3:25:DC:BC:DD:24:0E:A0:4D:AF:58:B4:7E:18:4

Alias name: geotrustprimarycertificationauthorityg3
Certificate fingerprints:
  MD5:  B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D

Alias name: geotrustprimarycertificationauthorityg2
Certificate fingerprints:
  MD5:  01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6

Alias name: godaddyclass2ca
Certificate fingerprints:
  MD5:  91:DE:06:25:AB:DA:FD:32:17:0C:BB:25:17:2A:84:67
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E

Alias name: trustcoreca1
Certificate fingerprints:
  MD5:  27:92:23:1D:0A:F5:40:7C:E9:E6:6B:9D:D8:F5:E7:6C
  SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
  SHA256:
 5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9

Alias name: hellenicacademicandresearchinstitutionseccrootca2015
Certificate fingerprints:
  MD5:  81:E5:B4:17:EB:C2:F5:E1:4B:0D:41:7B:49:92:FE:EF
  SHA1: 9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
  SHA256:
 44:B5:45:AA:8A:25:E6:5A:73:CA:15:DC:27:FC:36:D2:4C:1C:B9:95:3A:06:65:39:B1:15:82:DC:48:7B:48:3

Alias name: utnuserfirstobjectca
Certificate fingerprints:
  MD5:  A7:F2:E4:16:06:41:11:50:30:6B:9C:E3:B4:9C:B0:C9
```

```
  SHA1: E1:2D:FB:4B:41:D7:D9:C3:2B:30:51:4B:AC:1D:81:D8:38:5E:2D:46
  SHA256:
 6F:FF:78:E4:00:A7:0C:11:01:1C:D8:59:77:C4:59:FB:5A:F9:6A:3D:F0:54:08:20:D0:F4:B8:60:78:75:E5:8

Alias name: ttelesecglobalrootclass3
Certificate fingerprints:
  MD5:  CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B

Alias name: ttelesecglobalrootclass2
Certificate fingerprints:
  MD5:  2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5

Alias name: addtrustclass1ca
Certificate fingerprints:
  MD5:  1E:42:95:02:33:92:6B:B9:5F:C0:7F:DA:D6:B2:4B:FC
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A

Alias name: amzninternalrootca
Certificate fingerprints:
  MD5:  08:09:73:AC:E0:78:41:7C:0A:26:33:51:E8:CF:E6:60
  SHA1: A7:B7:F6:15:8A:FF:1E:C8:85:13:38:BC:93:EB:A2:AB:A4:09:EF:06
  SHA256:
 0E:DE:63:C1:DC:7A:8E:11:F1:AB:BC:05:4F:59:EE:49:9D:62:9A:2F:DE:9C:A7:16:32:A2:64:29:3E:8B:66:A

Alias name: starfieldrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F

Alias name: camerfirmachambersignca
Certificate fingerprints:
  MD5:  9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
```

```
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C

Alias name: secomscrootca2
Certificate fingerprints:
  MD5:  6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F

Alias name: entrustevca
Certificate fingerprints:
  MD5:  D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4

Alias name: secomscrootca1
Certificate fingerprints:
  MD5:  F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6

Alias name: affirmtrustcommercial
Certificate fingerprints:
  MD5:  82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6:B7
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A

Alias name: digicertassuredidrootg3
Certificate fingerprints:
  MD5:  7C:7F:65:31:0C:81:DF:8D:BA:3E:99:E2:5C:AD:6E:FB
  SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
  SHA256:
 7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C

Alias name: affirmtrustnetworking
Certificate fingerprints:
  MD5:  42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
```

```
Alias name: izenpecom
Certificate fingerprints:
  MD5:  A6:B0:CD:85:80:DA:5C:50:34:A3:39:90:2F:55:67:73
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1

Alias name: amazon-ca-g4-legacy
Certificate fingerprints:
  MD5:  6C:E5:BD:67:A4:4F:E3:FD:C2:4C:46:E6:06:5B:6D:55
  SHA1: EA:E7:DE:F9:0A:BE:9F:0B:68:CE:B7:24:0D:80:74:03:BF:6E:B1:6E
  SHA256:
 CD:72:C4:7F:B4:AD:28:A4:67:2B:E1:86:47:D4:40:E9:3B:16:2D:95:DB:3C:2F:94:BB:81:D9:09:F7:91:24:5

Alias name: digicertassuredidrootg2
Certificate fingerprints:
  MD5:  92:38:B9:F8:63:24:82:65:2C:57:33:E6:FE:81:8F:9D
  SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
  SHA256:
 7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:8

Alias name: comodoaaaservicesroot
Certificate fingerprints:
  MD5:  49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F

Alias name: entrustnetpremium2048secureserverca
Certificate fingerprints:
  MD5:  EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7

Alias name: trustcorrootcertca2
Certificate fingerprints:
  MD5:  A2:E1:F8:18:0B:BA:45:D5:C7:41:2A:BB:37:52:45:64
  SHA1: B8:BE:6D:CB:56:F1:55:B9:63:D4:12:CA:4E:06:34:C7:94:B2:1C:C0
  SHA256:
 07:53:E9:40:37:8C:1B:D5:E3:83:6E:39:5D:AE:A5:CB:83:9E:50:46:F1:BD:0E:AE:19:51:CF:10:FE:C7:C9:6

Alias name: entrust2048ca
```

```
Certificate fingerprints:
  MD5:  EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7

Alias name: trustcorrootcertca1
Certificate fingerprints:
  MD5:  6E:85:F1:DC:1A:00:D3:22:D5:B2:B2:AC:6B:37:05:45
  SHA1: FF:BD:CD:E7:82:C8:43:5E:3C:6F:26:86:5C:CA:A8:3A:45:5B:C3:0A
  SHA256:
 D4:0E:9C:86:CD:8F:E4:68:C1:77:69:59:F4:9E:A7:74:FA:54:86:84:B6:C4:06:F3:90:92:61:F4:DC:E2:57:5

Alias name: baltimorecybertrustroot
Certificate fingerprints:
  MD5:  AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E

Alias name: eecertificationcentrerootca
Certificate fingerprints:
  MD5:  43:5E:88:D4:7D:1A:4A:7E:FD:84:2E:52:EB:01:D4:6F
  SHA1: C9:A8:B9:E7:55:80:5E:58:E3:53:77:A7:25:EB:AF:C3:7B:27:CC:D7
  SHA256:
 3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09:79:CA:08:4E:46:85:68:1F:F1:95:CC:BA:8A:22:9B:8A:7

Alias name: dstacescax6
Certificate fingerprints:
  MD5:  21:D8:4C:82:2B:99:09:33:A2:EB:14:24:8D:8E:5F:E8
  SHA1: 40:54:DA:6F:1C:3F:40:74:AC:ED:0F:EC:CD:DB:79:D1:53:FB:90:1D
  SHA256:
 76:7C:95:5A:76:41:2C:89:AF:68:8E:90:A1:C7:0F:55:6C:FD:6B:60:25:DB:EA:10:41:6D:7E:B6:83:1F:8C:4

Alias name: comodocertificationauthority
Certificate fingerprints:
  MD5:  5C:48:DC:F7:42:72:EC:56:94:6D:1C:CC:71:35:80:75
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6

Alias name: thawteserverca
Certificate fingerprints:
  MD5:  EE:FE:61:69:65:6E:F8:9C:C6:2A:F4:D7:2B:63:EF:A2
```

```
  SHA1: 9F:AD:91:A6:CE:6A:C6:C5:00:47:C4:4E:C9:D4:A5:0D:92:D8:49:79
  SHA256:
 87:C6:78:BF:B8:B2:5F:38:F7:E9:7B:33:69:56:BB:CF:14:4B:BA:CA:A5:36:47:E6:1A:23:25:BC:10:55:31:6

Alias name: secomvalicertclass1ca
Certificate fingerprints:
  MD5:  65:58:AB:15:AD:57:6C:1E:A8:A7:B5:69:AC:BF:FF:EB
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0

Alias name: godaddyrootg2ca
Certificate fingerprints:
  MD5:  80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D

Alias name: globalchambersignroot2008
Certificate fingerprints:
  MD5:  9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C

Alias name: equifaxsecureebusinessca1
Certificate fingerprints:
  MD5:  14:C0:08:E5:A3:85:03:A3:BE:78:E9:67:4F:27:CA:EE
  SHA1: AE:E6:3D:70:E3:76:FB:C7:3A:EB:B0:A1:C1:D4:C4:7A:A7:40:B3:F4
  SHA256:
 2E:3A:2B:B5:11:25:05:83:6C:A8:96:8B:E2:CB:37:27:CE:9B:56:84:5C:6E:E9:8E:91:85:10:4A:FB:9A:F5:9

Alias name: quovadisrootca3
Certificate fingerprints:
  MD5:  31:85:3C:62:94:97:63:B9:AA:FD:89:4E:AF:6F:E0:CF
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3

Alias name: usertrustecccertificationauthority
Certificate fingerprints:
  MD5:  FA:68:BC:D9:B5:7F:AD:FD:C9:1D:06:83:28:CC:24:C1
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
```

```
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7

Alias name: quovadisrootca2
Certificate fingerprints:
  MD5:  5E:39:7B:DD:F8:BA:EC:82:E9:AC:62:BA:0C:54:00:2B
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8

Alias name: soneraclass2ca
Certificate fingerprints:
  MD5:  A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2

Alias name: twcarootcertificationauthority
Certificate fingerprints:
  MD5:  AA:08:8F:F6:F9:7B:B7:F2:B1:A7:1E:9B:EA:EA:BD:79
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4

Alias name: baltimorecybertrustca
Certificate fingerprints:
  MD5:  AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E

Alias name: cia-crt-g3-01-ca
Certificate fingerprints:
  MD5:  E3:66:DD:D6:A0:D5:40:8F:FF:29:E2:C0:CB:6E:62:1A
  SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
  SHA256:
 20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:E

Alias name: entrustrootcertificationauthorityg2
Certificate fingerprints:
  MD5:  4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
```

```
Alias name: verisignclass3g4ca
Certificate fingerprints:
  MD5:  3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7

Alias name: xrampglobalcaroot
Certificate fingerprints:
  MD5:  A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A

Alias name: identrustcommercialrootca1
Certificate fingerprints:
  MD5:  B3:3E:77:73:75:EE:A0:D3:E3:7E:49:63:49:59:BB:C7
  SHA1: DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
  SHA256:
 5D:56:49:9B:E4:D2:E0:8B:CF:CA:D0:8A:3E:38:72:3D:50:50:3B:DE:70:69:48:E4:2F:55:60:30:19:E5:28:A

Alias name: camerfirmachamberscommerceca
Certificate fingerprints:
  MD5:  B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C

Alias name: verisignclass3g2ca
Certificate fingerprints:
  MD5:  A2:33:9B:4C:74:78:73:D4:6C:E7:C1:F3:8D:CB:5C:E9
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8

Alias name: deutschetelekomrootca2
Certificate fingerprints:
  MD5:  74:01:4A:91:B1:08:C4:58:CE:47:CD:F0:DD:11:53:08
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D

Alias name: certumca
```

```
Certificate fingerprints:
  MD5:  2C:8F:9F:66:1D:18:90:B1:47:26:9D:8E:86:82:8C:A9
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2

Alias name: cybertrustglobalroot
Certificate fingerprints:
  MD5:  72:E4:4A:87:E3:69:40:80:77:EA:BC:E3:F4:FF:F0:E1
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A

Alias name: globalsignrootca
Certificate fingerprints:
  MD5:  3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9

Alias name: secomevrootca1
Certificate fingerprints:
  MD5:  22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3

Alias name: globalsignr3ca
Certificate fingerprints:
  MD5:  C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3

Alias name: staatdernederlandenrootcag3
Certificate fingerprints:
  MD5:  0B:46:67:07:DB:10:2F:19:8C:35:50:60:D1:0B:F4:37
  SHA1: D8:EB:6B:41:51:92:59:E0:F3:E7:85:00:C0:3D:B6:88:97:C9:EE:FC
  SHA256:
 3C:4F:B0:B9:5A:B8:B3:00:32:F4:32:B8:6F:53:5F:E1:72:C1:85:D0:FD:39:86:58:37:CF:36:18:7F:A6:F4:2

Alias name: staatdernederlandenrootcag2
Certificate fingerprints:
  MD5:  7C:A5:0F:F8:5B:9A:7D:6D:30:AE:54:5A:E3:42:A2:8A
```

```
  SHA1: 59:AF:82:79:91:86:C7:B4:75:07:CB:CF:03:57:46:EB:04:DD:B7:16
  SHA256:
 66:8C:83:94:7D:A6:3B:72:4B:EC:E1:74:3C:31:A0:E6:AE:D0:DB:8E:C5:B3:1B:E3:77:BB:78:4F:91:B6:71:6

Alias name: aolrootca2
Certificate fingerprints:
  MD5:  D6:ED:3C:CA:E2:66:0F:AF:10:43:0D:77:9B:04:09:BF
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B

Alias name: dstrootcax3
Certificate fingerprints:
  MD5:  41:03:52:DC:0F:F7:50:1B:16:F0:02:8E:BA:6F:45:C5
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3

Alias name: trustcenteruniversalcai
Certificate fingerprints:
  MD5:  45:E1:A5:72:C5:A9:36:64:40:9E:F5:E4:58:84:67:8C
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E

Alias name: aolrootca1
Certificate fingerprints:
  MD5:  14:F1:08:AD:9D:FA:64:E2:89:E7:1C:CF:A8:AD:7D:5E
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E

Alias name: affirmtrustpremiumecc
Certificate fingerprints:
  MD5:  64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2

Alias name: microseceszignorootca2009
Certificate fingerprints:
  MD5:  F8:49:F4:03:BC:44:2D:83:BE:48:69:7D:29:64:FC:B1
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
```

```
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7

Alias name: verisignclass1g3ca
Certificate fingerprints:
  MD5:  B1:47:BC:18:57:D1:18:A0:78:2D:EC:71:E8:2A:95:73
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6

Alias name: certplusrootcag2
Certificate fingerprints:
  MD5:  A7:EE:C4:78:2D:1B:EE:2D:B9:29:CE:D6:A7:96:32:31
  SHA1: 4F:65:8E:1F:E9:06:D8:28:02:E9:54:47:41:C9:54:25:5D:69:CC:1A
  SHA256:
 6C:C0:50:41:E6:44:5E:74:69:6C:4C:FB:C9:F8:0F:54:3B:7E:AB:BB:44:B4:CE:6F:78:7C:6A:99:71:C4:2F:1

Alias name: certplusrootcag1
Certificate fingerprints:
  MD5:  7F:09:9C:F7:D9:B9:5C:69:69:56:D5:37:3E:14:0D:42
  SHA1: 22:FD:D0:B7:FD:A2:4E:0D:AC:49:2C:A0:AC:A6:7B:6A:1F:E3:F7:66
  SHA256:
 15:2A:40:2B:FC:DF:2C:D5:48:05:4D:22:75:B3:9C:7F:CA:3E:C0:97:80:78:B0:F0:EA:76:E5:61:A6:C7:43:3

Alias name: addtrustexternalca
Certificate fingerprints:
  MD5:  1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F

Alias name: entrustrootcertificationauthority
Certificate fingerprints:
  MD5:  D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4

Alias name: verisignclass3ca
Certificate fingerprints:
  MD5:  EF:5A:F1:33:EF:F1:CD:BB:51:02:EE:12:14:4B:96:C4
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
```

```
Alias name: digicertassuredidrootca
Certificate fingerprints:
  MD5:  87:CE:0B:7B:2A:0E:49:00:E1:58:71:9B:37:A8:93:72
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5

Alias name: globalsignrootcar3
Certificate fingerprints:
  MD5:  C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3

Alias name: globalsignrootcar2
Certificate fingerprints:
  MD5:  94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9

Alias name: verisignclass1ca
Certificate fingerprints:
  MD5:  86:AC:DE:2B:C5:6D:C3:D9:8C:28:88:D3:8D:16:13:1E
  SHA1: CE:6A:64:A3:09:E4:2F:BB:D9:85:1C:45:3E:64:09:EA:E8:7D:60:F1
  SHA256:
 51:84:7C:8C:BD:2E:9A:72:C9:1E:29:2D:2A:E2:47:D7:DE:1E:3F:D2:70:54:7A:20:EF:7D:61:0F:38:B8:84:2

Alias name: thawtepremiumserverca
Certificate fingerprints:
  MD5:  A6:6B:60:90:23:9B:3F:2D:BB:98:6F:D6:A7:19:0D:46
  SHA1: E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2E:FC:66:66
  SHA256:
 3F:9F:27:D5:83:20:4B:9E:09:C8:A3:D2:06:6C:4B:57:D3:A2:47:9C:36:93:65:08:80:50:56:98:10:5D:BC:E

Alias name: verisigntsaca
Certificate fingerprints:
  MD5:  F2:89:95:6E:4D:05:F0:F1:A7:21:55:7D:46:11:BA:47
  SHA1: 20:CE:B1:F0:F5:1C:0E:19:A9:F3:8D:B1:AA:8E:03:8C:AA:7A:C7:01
  SHA256:
 CB:6B:05:D9:E8:E5:7C:D8:82:B1:0B:4D:B7:0D:E4:BB:1D:E4:2B:A4:8A:7B:D0:31:8B:63:5B:F6:E7:78:1A:9

Alias name: thawteprimaryrootca
```

```
Certificate fingerprints:
  MD5:  8C:CA:DC:0B:22:CE:F5:BE:72:AC:41:1A:11:A8:D8:12
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9

Alias name: visaecommerceroot
Certificate fingerprints:
  MD5:  FC:11:B8:D8:08:93:30:00:6D:23:F9:7E:EB:52:1E:02
  SHA1: 70:17:9B:86:8C:00:A4:FA:60:91:52:22:3F:9F:3E:32:BD:E0:05:62
  SHA256:
 69:FA:C9:BD:55:FB:0A:C7:8D:53:BB:EE:5C:F1:D5:97:98:9F:D0:AA:AB:20:A2:51:51:BD:F1:73:3E:E7:D1:2

Alias name: digicertglobalrootg3
Certificate fingerprints:
  MD5:  F5:5D:A4:50:A5:FB:28:7E:1E:0F:0D:CC:96:57:56:CA
  SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
  SHA256:
 31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D

Alias name: xrampglobalca
Certificate fingerprints:
  MD5:  A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A

Alias name: digicertglobalrootg2
Certificate fingerprints:
  MD5:  E4:A6:8A:C8:54:AC:52:42:46:0A:FD:72:48:1B:2A:44
  SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
  SHA256:
 CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5

Alias name: valicertclass2ca
Certificate fingerprints:
  MD5:  A9:23:75:9B:BA:49:36:6E:31:C2:DB:F2:E7:66:BA:87
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6

Alias name: geotrustprimaryca
Certificate fingerprints:
  MD5:  02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
```

```
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6

Alias name: netlockaranyclassgoldfotanusitvany
Certificate fingerprints:
  MD5:  C5:A1:B7:FF:73:DD:D6:D7:34:32:18:DF:FC:3C:AD:88
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9

Alias name: geotrustglobalca
Certificate fingerprints:
  MD5:  F7:75:AB:29:FB:51:4E:B7:77:5E:FF:05:3C:99:8E:F5
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3

Alias name: oistewisekeyglobalrootgbca
Certificate fingerprints:
  MD5:  A4:EB:B9:61:28:2E:B7:2F:98:B0:35:26:90:99:51:1D
  SHA1: 0F:F9:40:76:18:D3:D7:6A:4B:98:F0:A8:35:9E:0C:FD:27:AC:CC:ED
  SHA256:
 6B:9C:08:E8:6E:B0:F7:67:CF:AD:65:CD:98:B6:21:49:E5:49:4A:67:F5:84:5E:7B:D1:ED:01:9F:27:B8:6B:D

Alias name: certumtrustednetworkca2
Certificate fingerprints:
  MD5:  6D:46:9E:D9:25:6D:08:23:5B:5E:74:7D:1E:27:DB:F2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0

Alias name: starfieldservicesrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B

Alias name: comodorsacertificationauthority
Certificate fingerprints:
  MD5:  1B:31:B0:71:40:36:CC:14:36:91:AD:C4:3E:FD:EC:18
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
```

```
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3

Alias name: comodoaaaca
Certificate fingerprints:
  MD5:  49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F

Alias name: identrustpublicsectorrootca1
Certificate fingerprints:
  MD5:  37:06:A5:B0:FC:89:9D:BA:F4:6B:8C:1A:64:CD:D5:BA
  SHA1: BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
  SHA256:
 30:D0:89:5A:9A:44:8A:26:20:91:63:55:22:D1:F5:20:10:B5:86:7A:CA:E1:2C:78:EF:95:8F:D4:F4:38:9F:2

Alias name: certplusclass2primaryca
Certificate fingerprints:
  MD5:  88:2C:8C:52:B8:A2:3C:F3:F7:BB:03:EA:AE:AC:42:0B
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C

Alias name: ttelesecglobalrootclass2ca
Certificate fingerprints:
  MD5:  2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5

Alias name: accvraiz1
Certificate fingerprints:
  MD5:  D0:A0:5A:EE:05:B6:09:94:21:A1:7D:F1:B2:29:82:02
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1

Alias name: digicerthighassuranceevrootca
Certificate fingerprints:
  MD5:  D4:74:DE:57:5C:39:B2:D3:9C:85:83:C5:C0:65:49:8A
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
```

```
Alias name: amzninternalinfoseccag3
Certificate fingerprints:
  MD5:  E9:34:94:02:BA:BB:31:6B:22:E6:2B:A9:C4:F0:26:04
  SHA1: B9:B1:CA:38:F7:BF:9C:D2:D4:95:E7:B6:5E:75:32:9B:A8:78:2E:F6
  SHA256:
 81:03:0B:C7:E2:54:DA:7B:F8:B7:45:DB:DD:41:15:89:B5:A3:81:86:FB:4B:29:77:1F:84:0A:18:D9:67:6D:6

Alias name: cia-crt-g3-02-ca
Certificate fingerprints:
  MD5:  FD:B9:23:FD:D3:EB:2D:3E:57:EF:56:FF:DB:D3:E4:B9
  SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
  SHA256:
 93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C

Alias name: entrustrootcertificationauthorityec1
Certificate fingerprints:
  MD5:  B6:7E:1D:F0:58:C5:49:6C:24:3B:3D:ED:98:18:ED:BC
  SHA1: 20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
  SHA256:
 02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E:DF:F

Alias name: securitycommunicationrootca
Certificate fingerprints:
  MD5:  F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6

Alias name: globalsignca
Certificate fingerprints:
  MD5:  3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9

Alias name: trustcenterclass2caii
Certificate fingerprints:
  MD5:  CE:78:33:5C:59:78:01:6E:18:EA:B9:36:A0:B9:2E:23
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B

Alias name: camerfirmachambersofcommerceroot
```

```
Certificate fingerprints:
  MD5:  B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C

Alias name: geotrustprimarycag3
Certificate fingerprints:
  MD5:  B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D

Alias name: geotrustprimarycag2
Certificate fingerprints:
  MD5:  01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6

Alias name: hongkongpostrootca1
Certificate fingerprints:
  MD5:  A8:0D:6F:39:78:B9:43:6D:77:42:6D:98:5A:CC:23:CA
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B

Alias name: affirmtrustpremiumeccca
Certificate fingerprints:
  MD5:  64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2

Alias name: hellenicacademicandresearchinstitutionsrootca2015
Certificate fingerprints:
  MD5:  CA:FF:E2:DB:03:D9:CB:4B:E9:0F:AD:84:FD:7B:18:CE
  SHA1: 01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
  SHA256:
 A0:40:92:9A:02:CE:53:B4:AC:F4:F2:FF:C6:98:1C:E4:49:6F:75:5E:6D:45:FE:0B:2A:69:2B:CD:52:52:3F:3
```

IoT Analytics

L'action AWS IoT Analytics (`iotAnalytics`) envoie les données d'un message MQTT à un AWS IoT Analytics canal.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`iotanalytics:BatchPutMessage`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

  La politique attachée au rôle spécifié doit ressembler à l'exemple suivant.

  JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotanalytics:BatchPutMessage",
            "Resource": [
                "arn:aws:iotanalytics:us-west-2:account-id:channel/mychannel"
            ]
        }
    ]
}
```

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`batchMode`

(Facultatif) traiter ou non l'action en tant que lot. La valeur par défaut est `false`.

Lorsque `batchMode` c'est le cas `true` et que l'instruction SQL rule est évaluée à un tableau, chaque élément du tableau est délivré sous forme de message distinct lorsqu'il est transmis `BatchPutMessage`au AWS IoT Analytics canal. Le tableau résultant ne peut pas contenir plus de 100 messages.

Prend en charge les modèles de substitution : Non

`channelName`

Nom du AWS IoT Analytics canal sur lequel les données doivent être écrites.

Supporte les modèles de substitution : API et AWS CLI uniquement

`roleArn`

Rôle IAM qui permet d'accéder au AWS IoT Analytics canal. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

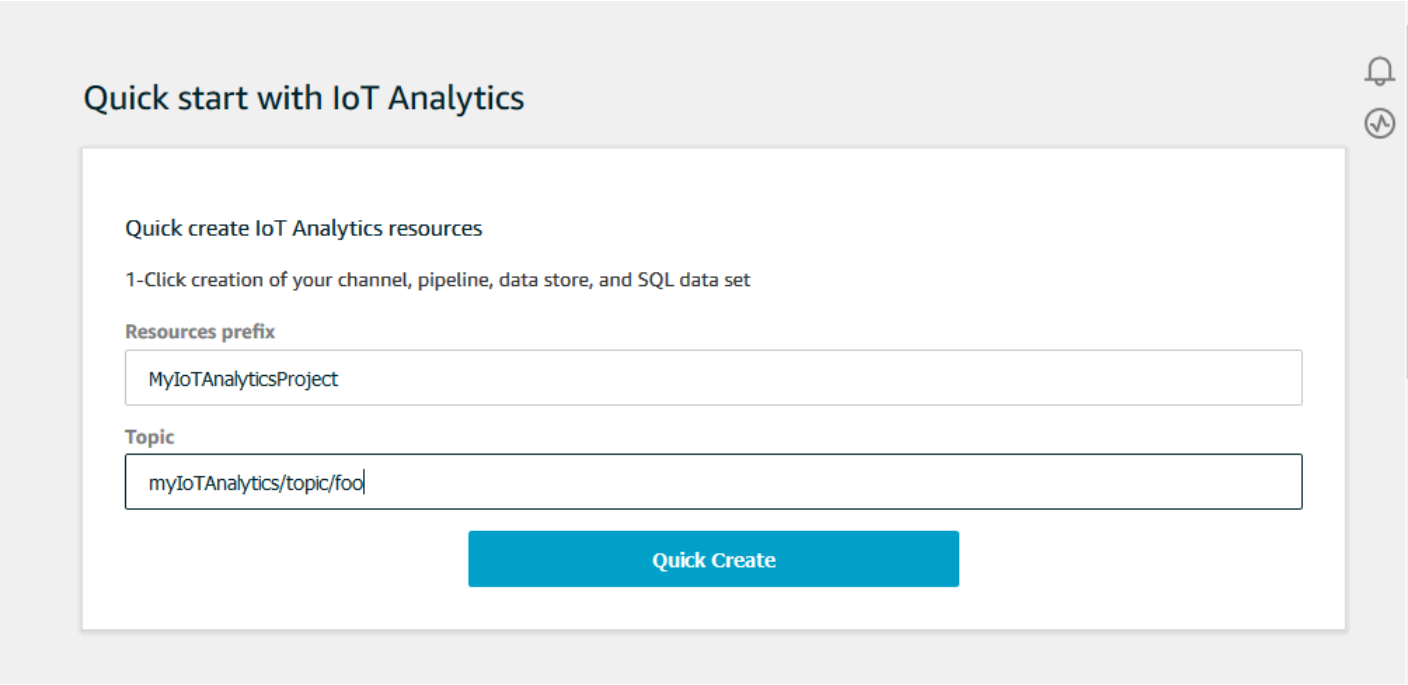
L'exemple JSON suivant définit une AWS IoT Analytics action dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotAnalytics": {
                    "channelName": "mychannel",
                    "roleArn": "arn:aws:iam::123456789012:role/analyticsRole",
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce que c'est AWS IoT Analytics ? dans le guide de AWS IoT Analytics l'utilisateur

- La AWS IoT Analytics console dispose également d'une fonction de démarrage rapide qui vous permet de créer un canal, un magasin de données, un pipeline et un magasin de données en un seul clic. Pour de plus amples informations, veuillez consulter AWS IoT Analytics console quickstart guide dans le AWS IoT Analytics Guide d'utilisateur.



AWS IoT Events

L'action AWS IoT Events (`iotEvents`) envoie les données d'un message MQTT à une AWS IoT Events entrée.

> **Important**
>
> Si la charge utile est envoyée AWS IoT Core sans le`Input attribute Key`, ou si la clé ne se trouve pas dans le même chemin JSON spécifié dans la clé, la règle IoT échouera avec l'erreur`Failed to send message to Iot Events`.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`iotevents:BatchPutMessage`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`batchMode`

(Optionnal) traiter ou non l'action d'événement en tant que lot. La valeur par défaut est `false`.

Lorsque `batchMode` c'est le cas `true` et que l'instruction SQL rule est évaluée à un tableau, chaque élément du tableau est traité comme un message distinct lorsqu'il est envoyé à AWS IoT Events par appel `BatchPutMessage`. Le tableau résultant ne peut pas contenir plus de 10 messages.

Quand `batchMode` est `true`, vous ne pouvez pas spécifier un `messageId`.

Prend en charge les modèles de substitution : Non

`inputName`

Le nom de l' AWS IoT Events entrée.

Supporte les modèles de substitution : API et AWS CLI uniquement

`messageId`

(Facultatif) Utilisez-le pour vérifier qu'une seule entrée (message) contenant une donnée `messageId` est traitée par un AWS IoT Events détecteur. Vous pouvez utiliser le modèle de `${newuuid()}` substitution pour générer un identifiant unique pour chaque demande.

Quand `batchMode` équivaut à `true`, vous ne pouvez pas spécifier de valeur `messageId` ; une nouvelle valeur UUID sera attribuée.

Prend en charge les modèles de substitution : Oui

`roleArn`

Rôle IAM qui permet d' AWS IoT envoyer une entrée à un AWS IoT Events détecteur. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant montre comment définir une action IoT Events dans une AWS IoT règle :

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotEvents": {
                    "inputName": "MyIoTEventsInput",
                    "messageId": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_events"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce que c'est AWS IoT Events ? dans le guide AWS IoT Events du développeur

AWS IoT SiteWise

L'action AWS IoT SiteWise (`iotSiteWise`) envoie les données d'un message MQTT aux propriétés des actifs dans AWS IoT SiteWise.

Vous pouvez suivre un didacticiel qui explique comment ingérer des données provenant d' AWS IoT objets. Pour plus d'informations, consultez le didacticiel Ingestion de données vers AWS IoT SiteWise des AWS IoT objets ou la section Ingestion de données à l'aide des règles de AWS IoT base du guide de l'AWS IoT SiteWise utilisateur.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`iotsitewise:BatchPutAssetPropertyValue`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Vous pouvez attacher l'exemple suivant de politique de confiance au rôle.

  JSON

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

  Pour améliorer la sécurité, vous pouvez spécifier un chemin hiérarchique des AWS IoT SiteWise actifs dans la `Condition` propriété. L'exemple suivant est une stratégie d'approbation qui spécifie un chemin de hiérarchie de ressource.

  JSON

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

```
                }
            }
        }
    ]
}
```

- Lorsque vous envoyez des données à AWS IoT SiteWise avec cette action, celles-ci doivent répondre aux exigences de l'`BatchPutAssetPropertyValue`opération. Pour plus d'informations, consultez BatchPutAssetPropertyValue dans la Référence d'API AWS IoT SiteWise .

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`putAssetPropertyValueEntries`

Une liste d'entrées de valeurs de propriétés de ressources dont chaque entrée contient les informations suivantes :

`propertyAlias`

(Facultatif) l'alias de propriété associé à votre propriété de ressources. Vous devez spécifier un `propertyAlias` ou à la fois un `assetId` et un `propertyId`. Pour plus d'informations sur les alias de propriété, veuillez consulter Mappage de flux de données industrielles en propriétés de ressource dans le Guide de l'utilisateur AWS IoT SiteWise .

Prend en charge les modèles de substitution : Oui

`assetId`

(Facultatif) L'ID de l' AWS IoT SiteWise actif. Vous devez spécifier un `propertyAlias` ou à la fois un `assetId` et un `propertyId`.

Prend en charge les modèles de substitution: Oui

`propertyId`

(Facultatif) ID d'une propriété d'actif. Vous devez spécifier un `propertyAlias` ou à la fois un `assetId` et un `propertyId`.

Prend en charge les modèles de substitution: Oui

`entryId`

(Facultatif) Un identifiant unique pour cette entrée. Définir le `entryId` pour mieux savoir quel message a provoqué une erreur en cas de défaillance. La valeur par défaut est un nouvel UUID.

Prend en charge les modèles de substitution : Oui

`propertyValues`

Liste des valeurs de propriété à insérer, contenant chacune l'horodatage, la qualité et la valeur (TQV) au format suivant :

`timestamp`

Structure d'horodatage contenant les informations suivantes :

`timeInSeconds`

Chaîne contenant l'heure en secondes au format d'heure Unix epoch. Si votre charge utile de message n'a pas d'horodatage, vous pouvez utiliser timestamp(), qui renvoie l'heure actuelle en millisecondes. Pour convertir cette heure en secondes, vous pouvez utiliser le modèle de substitution suivant : **`${floor(timestamp() / 1E3)}`**.

Prend en charge les modèles de substitution : Oui

`offsetInNanos`

(Facultatif) Chaîne contenant le décalage de nanoseconde par rapport à l'heure en secondes. Si votre charge utile de message n'a pas d'horodatage, vous pouvez utiliser timestamp(), qui renvoie l'heure actuelle en millisecondes. Pour calculer le décalage de nanosecondes à partir de cette heure, vous pouvez utiliser le modèle de substitution suivant : **`${(timestamp() % 1E3) * 1E6}`**.

Prend en charge les modèles de substitution : Oui

En ce qui concerne Unix Epoch Time, n' AWS IoT SiteWise accepte que les entrées dont l'horodatage est compris entre 7 jours et 5 minutes dans le futur.

`quality`

(Facultatif) Chaîne qui décrit la qualité de la valeur. Valeurs valides : `GOOD`, `BAD`, `UNCERTAIN`.

Prend en charge les modèles de substitution : Oui

`value`

Structure de valeur qui contient l'un des champs de valeur suivants, en fonction du type de données de la propriété de la ressource :

`booleanValue`

(Facultatif) Chaîne qui contient la valeur booléenne de l'entrée de valeur.

Prend en charge les modèles de substitution : Oui

`doubleValue`

(Facultatif) Chaîne qui contient la valeur double de l'entrée de valeur.

Prend en charge les modèles de substitution : Oui

`integerValue`

(Facultatif) Chaîne qui contient la valeur d'entier de l'entrée de valeur.

Prend en charge les modèles de substitution : Oui

`stringValue`

(Facultatif) Valeur de chaîne de l'entrée de valeur.

Prend en charge les modèles de substitution : Oui

`roleArn`

L'ARN du rôle IAM qui AWS IoT autorise l'envoi d'une valeur de propriété d'actif à AWS IoT SiteWise. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une SiteWise action IoT de base dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
```

```
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "/some/property/alias",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${my.payload.timeInSeconds}"
                                    },
                                    "value": {
                                        "integerValue": "${my.payload.value}"
                                    }
                                }
                            ]
                        }
                    ],
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une SiteWise action IoT dans une AWS IoT règle. Cet exemple utilise la rubrique comme alias de propriété et la fonction `timestamp()`. Par exemple, si vous publiez des données vers `/company/windfarm/3/turbine/7/rpm`, cette action envoie les données à la propriété de ressource avec un alias de propriété identique à la rubrique que vous avez spécifiée.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "${topic()}",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${floor(timestamp() / 1E3)}",
```

```
                                    "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                                },
                                "value": {
                                    "doubleValue": "${my.payload.value}"
                                }
                            }
                        ]
                    }
                ],
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
            }
        }
    ]
  }
}
```

Consultez aussi

- Qu'est-ce que AWS IoT SiteWise ? dans le Guide de l'utilisateur AWS IoT SiteWise
- Ingestion de données à l'aide AWS IoT Core des règles du guide de l'AWS IoT SiteWise utilisateur
- Ingestion de données vers AWS IoT SiteWise des AWS IoT objets figurant dans le guide de l'AWS IoT SiteWise utilisateur
- Résolution des problèmes liés à une action de AWS IoT SiteWise règle dans le guide de AWS IoT SiteWise l'utilisateur

Firehose

L'action Firehose (`firehose`) envoie les données d'un message MQTT vers un flux Amazon Data Firehose.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`firehose:PutRecord`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

- Si vous utilisez Firehose pour envoyer des données vers un compartiment Amazon S3 et que vous utilisez un AWS KMS client géré AWS KMS key pour chiffrer les données au repos dans Amazon S3, Firehose doit avoir accès à votre compartiment et être autorisé à l'utiliser au nom de l'appelant AWS KMS key . Pour plus d'informations, consultez Accorder à Firehose l'accès à une destination Amazon S3 dans le manuel Amazon Data Firehose Developer Guide.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`batchMode`

(Facultatif) S'il faut fournir le flux Firehose sous forme de lot en utilisant. `PutRecordBatch` La valeur par défaut est `false`.

Quand `batchMode` est `true` et que l'instruction SQL de la règle est évaluée dans un tableau, chaque élément du tableau forme un enregistrement dans la demande `PutRecordBatch`. Le tableau résultant ne peut pas contenir plus de 500 enregistrements.

Prend en charge les modèles de substitution : Non

`deliveryStreamName`

Le flux Firehose dans lequel les données du message doivent être écrites.

Supporte les modèles de substitution : API et AWS CLI uniquement

`separator`

(Facultatif) Séparateur de caractères utilisé pour séparer les enregistrements écrits dans le flux Firehose. Si vous omettez ce paramètre, le flux n'utilise aucun séparateur. Valeurs valides : `,` (virgule), `\t` (tab), `\n` (nouvelle ligne), `\r\n` (nouvelle ligne Windows).

Prend en charge les modèles de substitution : Non

`roleArn`

Rôle IAM qui permet d'accéder au flux Firehose. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action Firehose dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "my_firehose_stream",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action Firehose avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon Data Firehose ? dans le guide du développeur Amazon Data Firehose

Kinesis Data Streams

L'action Kinesis Data Streams `kinesis` () écrit les données d'un message MQTT dans Amazon Kinesis Data Streams.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`kinesis:PutRecord`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez une clé KMS AWS KMS gérée par le client AWS KMS key pour chiffrer des données inactives dans Kinesis Data Streams, le service doit être autorisé à l'utiliser au nom de AWS KMS key l'appelant. Pour de plus amples informations, veuillez consulter Permissions to use user-generated AWS KMS keys dans le Amazon Kinesis Data Streams Developer Guide.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`stream`

Le Flux de donnees Kinesis dans lequel écrire les données.

Supporte les modèles de substitution : API et AWS CLI uniquement

`partitionKey`

Clé de partition utilisée pour déterminer dans quelle partition les données sont écrites. La clé de partition est généralement composée d'une expression (par exemple, `${topic()}` ou `${timestamp()}`).

Prend en charge les modèles de substitution : Oui

`roleArn`

L'ARN du rôle IAM qui accorde l' AWS IoT autorisation d'accéder au flux de données Kinesis. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action Kinesis Data Streams dans AWS IoT une règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "my_kinesis_stream",
                    "partitionKey": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action Kinesis avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "${topic()}",
                    "partitionKey": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
```

```
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce que Amazon Kinesis Data Streams ? dans le Guide du développeur Amazon Kinesis Data Streams

Lambda

Une action Lambda (`lambda`) appelle une AWS Lambda fonction en transmettant un message MQTT. AWS IoT invoque les fonctions Lambda de manière asynchrone.

Vous pouvez suivre un didacticiel qui vous montre comment créer et tester une règle avec une action Lambda. Pour de plus amples informations, veuillez consulter Tutoriel : Formatage d'une notification à l'aide d'une AWS Lambda fonction.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- AWS IoT Pour appeler une fonction Lambda, vous devez configurer une politique qui accorde l'`lambda:InvokeFunction`autorisation de. AWS IoT Vous ne pouvez appeler une fonction Lambda définie dans la même fonction que Région AWS lorsque votre politique Lambda existe. Les fonctions Lambda utilisent des politiques basées sur les ressources, vous devez donc attacher la politique à la fonction Lambda elle-même.

  Utilisez la AWS CLI commande suivante pour joindre une politique qui accorde l'`lambda:InvokeFunction`autorisation. Dans cette commande, remplacez :

  - *`function_name`*avec le nom de la fonction Lambda. Vous ajoutez une nouvelle autorisation pour mettre à jour la politique de ressources de la fonction.
  - *`region`*avec le Région AWS de la fonction.
  - *`account-id`*avec le Compte AWS numéro où la règle est définie.
  - *`rule-name`*avec le nom de la AWS IoT règle pour laquelle vous définissez l'action Lambda.
  - *`unique_id`*avec un identifiant de relevé unique.

> **Important**
>
> Si vous ajoutez une autorisation pour un AWS IoT principal sans fournir le `source-arn` ou`source-account`, toute autorisation Compte AWS qui crée une règle avec votre action Lambda peut activer des règles à partir desquelles appeler votre fonction Lambda. AWS IoT

Pour plus d'informations, consultez Autorisations AWS Lambda.

```
aws lambda add-permission \
    --function-name function_name \
    --region region \
    --principal iot.amazonaws.com \
    --source-arn arn:aws:iot:region:account-id:rule/rule_name \
    --source-account account-id
    --statement-id unique_id
    --action "lambda:InvokeFunction"
```

- Si vous utilisez la AWS IoT console pour créer une règle pour l'action de règle Lambda, la fonction Lambda est déclenchée automatiquement. Si vous utilisez AWS CloudFormation plutôt avec le `AWS::IoT::TopicRule LambdaAction`, vous devez ajouter une `AWS::lambda::Permission`ressource. La ressource vous autorise ensuite à déclencher la fonction Lambda.

  Le code suivant montre un exemple d'ajout de cette ressource. Dans cet exemple, remplacez :

  - *`function_name`*avec le nom de la fonction Lambda.
  - *`region`*avec le Région AWS de la fonction.
  - *`account-id`*avec le Compte AWS numéro où la règle est définie.
  - *`rule-name`*avec le nom de la AWS IoT règle pour laquelle vous définissez l'action Lambda.

```
Type: AWS::Lambda::Permission
Properties:
  Action: lambda:InvokeFunction
  FunctionName: !Ref function_name
  Principal: "iot.amazonaws.com"
  SourceAccount: account-id
  SourceArn: arn:aws:iot:region:account-id:rule/rule_name
```

- Si vous utilisez un AWS KMS client géré AWS KMS key pour chiffrer des données au repos dans Lambda, le service doit être autorisé à l'utiliser au nom AWS KMS key de l'appelant. Pour plus d'informations, veuillez consulter la rubrique Chiffrement au repos dans le Guide du développeur AWS Lambda .

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`functionArn`

L'ARN de la fonction Lambda à invoquer. AWS IoT doit être autorisé à invoquer la fonction. Pour de plus amples informations, veuillez consulter Prérequis.

Si vous ne spécifiez pas de version ou d'alias pour votre fonction Lambda, c'est la version la plus récente de la fonction qui est arrêtée. Vous pouvez spécifier une version ou un alias si vous souhaitez arrêter une version spécifique de votre fonction Lambda. Pour spécifier une version ou un alias, ajoutez la version ou l'alias à l'ARN de la fonction Lambda.

```
arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias
```

Pour plus d'informations sur le versionnage et les alias, consultez AWS Lambda la fonction versionnage et alias.

Supporte les modèles de substitution : API et AWS CLI uniquement

Exemples

L'exemple JSON suivant définit une action Lambda dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
```

```
                    "functionArn": "arn:aws:lambda:us-
east-2:123456789012:function:myLambdaFunction"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action Lambda avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:
${topic()}"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce que c'est AWS Lambda ? dans le guide AWS Lambda du développeur
- Tutoriel : Formatage d'une notification à l'aide d'une AWS Lambda fonction

Emplacement

L'action Location (`location`) envoie vos données de localisation géographique à Amazon Location Service.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`geo:BatchUpdateDevicePosition`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`deviceId`

ID unique de l'appareil fournissant les données de localisation. Pour plus d'informations, veuillez consulter la `DeviceId` de l' Amazon Location Service API Reference.

Prend en charge les modèles de substitution : Oui

`latitude`

Chaîne qui correspond à une valeur double représentant la latitude de l'emplacement de l'appareil.

Prend en charge les modèles de substitution : Oui

`longitude`

Chaîne qui correspond à une valeur double représentant la longitude de l'emplacement de l'appareil.

Prend en charge les modèles de substitution : Oui

`roleArn`

Rôle IAM qui autorise l'accès au domaine Amazon Location Service. Pour de plus amples informations, veuillez consulter Prérequis.

`timestamp`

Heure à laquelle les données de localisation ont été échantillonnées. La valeur par défaut est l'heure à laquelle le message MQTT a été traité.

La `timestamp` valeur se compose des deux valeurs suivantes :

- `value`: Expression qui renvoie une valeur temporelle de longue époque. Vous pouvez utiliser cette the section called “time_to_epoch (Chaîne, Chaîne)” fonction pour créer un horodatage valide à partir d'une valeur de date ou d'heure transmise dans la charge utile du message. Prend en charge les modèles de substitution : Oui
- `unit`: (Facultatif) La précision de la valeur d'horodatage résultant de l'expression décrite à la section `value`. Valeurs valides: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS` L'argument par défaut est `MILLISECONDS`. Supporte les modèles de substitution : API et AWS CLI uniquement.

`trackerName`

Nom de la ressource de suivi Amazon Location dans laquelle l'emplacement est mis à jour. Pour plus d'informations, consultez Tracker dans le guide du développeur d'Amazon Location Service.

Supporte les modèles de substitution : API et AWS CLI uniquement

Exemples

L'exemple JSON suivant définit une action de localisation dans une AWS IoT règle.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123454962127:role/service-role/ExampleRole",
     "trackerName": "MyTracker",
     "deviceId": "001",
     "sampleTime": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "-12.3456",
     "longitude": "65.4321"
    }
   }
  ]
 }
```

```
}
```

L'exemple JSON suivant définit une action de localisation avec des modèles de substitution dans une AWS IoT règle.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123456789012:role/service-role/ExampleRole",
     "trackerName": "${TrackerName}",
     "deviceId": "${DeviceID}",
     "timestamp": {
      "value": "${timestamp()}",
      "unit": "MILLISECONDS"
     },
     "latitude": "${get(position, 0)}",
     "longitude": "${get(position, 1)}"
    }
   }
  ]
 }
}
```

L'exemple de charge utile MQTT suivant montre comment les modèles de substitution de l'exemple précédent accèdent aux données. Vous pouvez utiliser la commande get-device-position-historyCLI pour vérifier que les données de charge utile MQTT sont fournies dans votre outil de localisation.

```
{
 "TrackerName": "mytracker",
 "DeviceID": "001",
 "position": [
  "-12.3456",
  "65.4321"
 ]
}
```

```
aws location get-device-position-history --device-id 001 --tracker-name mytracker
```

```
{
 "DevicePositions": [
  {
   "DeviceId": "001",
   "Position": [
    -12.3456,
    65.4321
   ],
   "ReceivedTime": "2022-11-11T01:31:54.464000+00:00",
   "SampleTime": "2022-11-11T01:31:54.308000+00:00"
  }
 ]
}
```

Consultez aussi

- Qu'est-ce qu'Amazon Location Service ? dans le guide du développeur d'Amazon Location Service.

OpenSearch

L'action OpenSearch (`openSearch`) écrit les données des messages MQTT dans un domaine Amazon OpenSearch Service. Vous pouvez ensuite utiliser des outils tels que OpenSearch les tableaux de bord pour interroger et visualiser les données dans OpenSearch Service.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`es:ESHttpPut`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez un client géré AWS KMS key pour chiffrer les données au repos dans le OpenSearch Service, le service doit être autorisé à utiliser la clé KMS au nom de l'appelant.

Pour plus d'informations, consultez la section Chiffrement des données au repos pour Amazon OpenSearch Service dans le manuel Amazon OpenSearch Service Developer Guide.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`endpoint`

Le point de terminaison de votre domaine Amazon OpenSearch Service.

Supporte les modèles de substitution : API et AWS CLI uniquement

`index`

L' OpenSearch index dans lequel vous souhaitez stocker vos données.

Prend en charge les modèles de substitution : Oui

`type`

Type de document que vous stockez.

> **Note**
>
> Pour OpenSearch les versions ultérieures à 1.0, la valeur du `type` paramètre doit être`_doc`. Pour plus d'informations, consultez la documentation OpenSearch .

Prend en charge les modèles de substitution : Oui

`id`

Identifiant unique de chaque document.

Prend en charge les modèles de substitution : Oui

`roleARN`

Rôle IAM qui permet d'accéder au domaine de OpenSearch service. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Limites

L'action OpenSearch (`openSearch`) ne peut pas être utilisée pour fournir des données aux clusters VPC Elasticsearch.

Exemples

L'exemple JSON suivant définit une OpenSearch action dans une AWS IoT règle et indique comment vous pouvez spécifier les champs de cette `OpenSearch` action. Pour de plus amples informations, veuillez consulter OpenSearchAction.

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "_doc",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une OpenSearch action avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
```

```
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

> **Note**
>
> Le `type` champ substitué fonctionne pour OpenSearch la version 1.0. Pour toutes les versions ultérieures à 1.0, la valeur de `type` doit être`_doc`.

Consultez aussi

Qu'est-ce qu'Amazon OpenSearch Service ? dans le Amazon OpenSearch Service Developer Guide

Republish

L'action republish (`republish`) republie un message MQTT dans un autre sujet MQTT.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`iot:Publish`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`headers`

Informations sur les en-têtes MQTT version 5.0.

Pour plus d'informations, veuillez consulter les sections RepublishAction et MqttHeaders (français non garanti) de la Référence d'API AWS .

`topic`

Rubrique MQTT dans laquelle republier le message.

Pour republier dans un sujet réservé, qui commence par$, utilisez $$ plutôt. Par exemple, si vous republiez sur un device shadow une rubrique du type `$aws/things/MyThing/shadow/update`, spécifiez la rubrique comme `$$aws/things/MyThing/shadow/update`.

> **Note**
>
> La republication vers des thèmes de la tâche réservés n'est pas prise en charge. AWS IoT Device Defender les rubriques réservées ne prennent pas en charge la publication HTTP.

Prend en charge les modèles de substitution : Oui

`qos`

(facultatif) Le niveau de qualité de service (QoS) à utiliser pour republier des messages. Valeurs valides : `0`, `1`. La valeur par défaut est `0`. Pour plus d'informations sur MQTT QoS, consultez MQTT.

Prend en charge les modèles de substitution : Non

`roleArn`

Rôle IAM qui permet de AWS IoT publier sur le sujet MQTT. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action de republication dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "another/topic",
                    "qos": 1,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action de republication avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action de republication `headers` dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
```

```
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
                    "headers": {
                        "payloadFormatIndicator": "UTF8_DATA",
                        "contentType": "rule/contentType",
                        "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
                        "userProperties": [
                            {
                                "key": "ruleKey1",
                                "value": "ruleValue1"
                            },
                            {
                                "key": "ruleKey2",
                                "value": "ruleValue2"
                            }
                        ]
                    }
                }
            }
        ]
    }
}
```

> **Note**
>
> L'adresse IP source d'origine ne sera pas transmise par le biais de Republier l'action.

S3

L'action S3 (`s3`) écrit les données d'un message MQTT dans un compartiment Amazon Simple Storage Service (Amazon S3).

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`s3:PutObject`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez un service AWS KMS géré par le client AWS KMS key pour chiffrer des données au repos dans Amazon S3, le service doit être autorisé à l'utiliser au nom AWS KMS key de l'appelant. Pour plus d'informations, consultez les sections « AWS géré » AWS KMS keys et « géré par le client » AWS KMS keys dans le manuel Amazon Simple Storage Service Developer Guide.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`bucket`

Compartiment Amazon S3 dans lequel écrire les données..

Supporte les modèles de substitution : API et AWS CLI uniquement

`cannedacl`

Liste ACL prédéfinie Amazon S3 qui contrôle l'accès à l'objet identifié par la clé d'objet. Pour plus d'informations, y compris les valeurs autorisées, voir Canned ACL.

Prend en charge les modèles de substitution :Non

`key`

Chemin d'accès au fichier dans lequel les données sont écrites.

Prenons un exemple où ce paramètre se trouve `${topic()}/${timestamp()}` et où la règle reçoit un message contenant le sujet `some/topic`. Si l'horodatage actuel est le cas`1460685389`, cette action écrit les données dans un fichier appelé `1460685389` dans le `some/topic` dossier du compartiment S3.

> **Note**
>
> Si vous utilisez une clé statique, elle AWS IoT remplace un seul fichier à chaque fois que la règle est invoquée. Nous vous recommandons d'utiliser l'horodatage du message ou

un autre identifiant de message unique afin qu'un nouveau fichier soit enregistré dans Amazon S3 pour chaque message reçu.

Prend en charge les modèles de substitution : Oui

`roleArn`

Rôle IAM qui autorise l'accès au compartiment Amazon S3. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action S3 dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "bucketName": "amzn-s3-demo-bucket",
                    "cannedacl": "public-read",
                    "key": "${topic()}/${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.

Salesforce IoT

Une action Salesforce IoT (`salesforce`) envoie les données du message MQTT qui a déclenché la règle pour un flux d'entrée Salesforce IoT.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`url`

URL exposée par le flux d'entrée Salesforce IoT. L'URL est disponible depuis la plateforme Salesforce IoT au moment où vous créez un flux d'entrée. Pour plus d'informations, consultez la documentation Salesforce IoT.

Prend en charge les modèles de substitution : Non

`token`

Jeton utilisé pour authentifier l'accès au flux d'entrée Salesforce IoT spécifié. Le jeton est disponible depuis la plateforme Salesforce IoT au moment où vous créez un flux d'entrée. Pour plus d'informations, consultez la documentation Salesforce IoT.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action Salesforce IoT dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "salesforce": {
                    "token": "ABCDEFGHI123456789abcdefghi123456789",
                    "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/
stream-id/connection-id/my-event"
                }
```

```
            }
        ]
    }
}
```

SNS

L'action SNS (`sns`) envoie les données d'un message MQTT sous la forme d'une notification push Amazon Simple Notification Service (Amazon SNS).

Vous pouvez suivre un didacticiel qui vous montre comment créer et tester une règle avec une action SNS. Pour de plus amples informations, veuillez consulter Tutoriel : Envoi d'une SNS notification Amazon.

> **Note**
>
> L'action SNS ne prend pas en charge les rubriques Amazon SNS FIFO (premier entré, premier sorti). Le moteur de règles étant un service entièrement distribué, l'ordre des messages n'est pas garanti lorsque l'action SNS est invoquée.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`sns:Publish`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez un AWS KMS service géré par le client AWS KMS key pour chiffrer des données au repos dans Amazon SNS, le service doit être autorisé à l'utiliser au nom de l'appelant AWS KMS key . Pour en savoir plus, veuillez consulter Key management dans le Guide du développeur Amazon Simple Notification Service.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`targetArn`

Rubrique SNS ou appareil individuel auxquels les notifications push sont envoyées.

Supporte les modèles de substitution : API et AWS CLI uniquement

`messageFormat`

(Facultatif) Format du message. Amazon SNS utilise ce paramètre pour déterminer si la charge utile doit être analysée et si les parties de la charge utile spécifiques à la plate-forme doivent être extraites. Valeurs valides : `JSON`, `RAW`. La valeur par défaut est `RAW`.

Prend en charge les modèles de substitution : Non

`roleArn`

Rôle IAM qui autorise l'accès à SNS. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action SNS dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-2:123456789012:my_sns_topic",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action SNS avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-1:123456789012:${topic()}",
                    "messageFormat": "JSON",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce qu'Amazon Simple Notification Service ? dans le Guide du développeur Amazon Simple Notification Service
- Tutoriel : Envoi d'une SNS notification Amazon

SQS

L'action SQS (`sqs`) envoie les données d'un message MQTT à une file d'attente Amazon Simple Queue Service (Amazon SQS).

> **Note**
>
> L'action SQS ne prend pas en charge les files d'attente Amazon SQS FIFO (premier entré, premier sorti). Le moteur de règles étant un service entièrement distribué, l'ordre des messages n'est pas garanti lorsque l'action SQS est déclenchée.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`sqs:SendMessage`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.
- Si vous utilisez un AWS KMS client géré AWS KMS key pour chiffrer des données au repos dans Amazon SQS, le service doit être autorisé à l'utiliser au nom AWS KMS key de l'appelant. Pour de plus amples informations, veuillez consulter Key management dans le Guide du développeur Amazon Simple Storage Service.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`queueUrl`

URL de la file d'attente Amazon SQS dans laquelle les données doivent être écrites. La région indiquée dans cette URL ne doit pas nécessairement être identique à celle de votre Région AWS AWS IoT règle.

> **Note**
>
> Des frais supplémentaires peuvent être facturés pour le transfert de données croisé Régions AWS à l'aide de l'action de règle SQS. Pour plus d'informations, consultez la tarification d'Amazon SQS.

Supporte les modèles de substitution : API et AWS CLI uniquement

`useBase64`

Définissez ce paramètre sur `true` pour configurer l'action de la règle afin de coder les données du message en base64 avant de les écrire dans la file d'attente Amazon SQS. La valeur par défaut est `false`.

Prend en charge les modèles de substitution : Non

`roleArn`

Le rôle IAM qui autorise l'accès à la file d'attente Amazon SQS. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action SQS dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
my_sqs_queue",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

L'exemple JSON suivant définit une action SQS avec des modèles de substitution dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
${topic()}",
```

```
                    "useBase64": true,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

Consultez aussi

- Présentation d'Amazon Simple Queue Service dans le Manuel du développeur Amazon Simple Queue Service

Step Functions

L'action Step Functions (`stepFunctions`) démarre une machine à AWS Step Functions états.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer la réalisation de l'`states:StartExecution`opération. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir ou créer un rôle pour autoriser l'exécution AWS IoT de cette action de règle.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`stateMachineName`

Le nom de la machine d'état Fonctions d'étape à démarrer.

Supporte les modèles de substitution : API et AWS CLI uniquement

`executionNamePrefix`

(Facultatif) Le nom donné à l'exécution de la machine d'état se compose de ce préfixe suivi d'un UUID. L'étape Fonctions crée un nom unique pour l'exécution de chaque machine d'état s'il n'y en a pas.

Prend en charge les modèles de substitution : Oui

`roleArn`

L'ARN du rôle qui AWS IoT autorise le démarrage de la machine à états. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

Exemples

L'exemple JSON suivant définit une action Step Functions dans une AWS IoT règle.

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "stepFunctions": {
                    "stateMachineName": "myStateMachine",
                    "executionNamePrefix": "myExecution",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_step_functions"
                }
            }
        ]
    }
}
```

Consultez aussi

- Qu'est-ce que c'est AWS Step Functions ? dans le guide AWS Step Functions du développeur

Timestream

L'action de règle Timestream écrit les attributs (mesures) d'un message MQTT dans une table Amazon Timestream. Pour plus d'informations sur Amazon Timestream, veuillez consulter Qu'est-ce qu'Amazon Timestream ?.

> **Note**
>
> Amazon Timestream n'est pas disponible dans tous les s. Région AWS Si Amazon Timestream n'est pas disponible dans votre région, il n'apparaîtra pas dans la liste des actions relatives aux règles.

Les attributs que cette règle stocke dans la base de données Timestream sont ceux qui résultent de l'instruction de requête de la règle. La valeur de chaque attribut dans le résultat de l'instruction de requête est analysée pour en déduire le type de données (comme dans le cas d'une the section called “Dynamo DBv2” action). La valeur de chaque attribut est écrite dans son propre enregistrement dans la table Timestream. Pour spécifier ou modifier le type de données d'un attribut, utilisez la `cast()`fonction contenue dans l'instruction de requête. Pour de plus amples informations sur le contenu de chaque enregistrement Timestream, veuillez consulter. the section called “Contenu d'un enregistrement Timestream”

> **Note**
>
> Avec SQL V2 (2016-03-23), les valeurs numériques qui sont des nombres entiers, telles que`10.0`, sont converties en leur représentation entière (). `10` Leur conversion explicite en une `Decimal` valeur, par exemple en utilisant la fonction cast (), n'empêche pas ce comportement : le résultat reste une `Integer` valeur. Cela peut provoquer des erreurs de non-concordance de type qui empêchent l'enregistrement des données dans la base de données Timestream. Pour traiter des valeurs numériques entières en tant que `Decimal` valeurs, utilisez SQL V1 (08/10/2015) pour l'instruction de requête de règle.

> **Note**
>
> Le nombre maximum de valeurs qu'une action de règle Timestream peut écrire dans une table Amazon Timestream est de 100. Pour de plus amples informations, veuillez consulter la Référence d'Amazon Timestream Quota.

Prérequis

Cette action réglementaire est assortie des exigences suivantes :

- Rôle IAM qui AWS IoT peut assumer l'exécution des `timestream:WriteRecords` opérations `timestream:DescribeEndpoints` et. Pour de plus amples informations, veuillez consulter Accorder à une AWS IoT règle l'accès dont elle a besoin.

  Dans la AWS IoT console, vous pouvez choisir, mettre à jour ou créer un rôle AWS IoT pour autoriser l'exécution de cette action de règle.
- Si vous utilisez un client AWS KMS pour chiffrer des données inactives dans Timestream, le service doit être autorisé à les utiliser au nom de l'appelant. AWS KMS key Pour plus d'informations, consultez Comment les AWS services utilisent AWS KMS.

Paramètres

Lorsque vous créez une AWS IoT règle avec cette action, vous devez spécifier les informations suivantes :

`databaseName`

Nom d'une base de données Amazon Timestream qui possède la table destinée à recevoir les enregistrements créés par cette action. Voir aussi **`tableName`**.

Supporte les modèles de substitution : API et AWS CLI uniquement

`dimensions`

Attributs de métadonnées de la série chronologique qui sont écrits dans chaque enregistrement de mesure. Par exemple, le nom et la zone de disponibilité d'une EC2 instance ou le nom du fabricant d'une éolienne sont des dimensions.

`name`

Nom de la dimension de métadonnées. Il s'agit du nom de la colonne dans l'enregistrement de table de base de données.

Les dimensions ne peuvent pas être nommées : `measure_namemeasure_value`, `outime`. Ces alias sont réservés. Les noms de dimension ne peuvent pas commencer par `ts_` ou `measure_value` ne peuvent pas contenir le caractère deux-points (`:`).

Prend en charge les modèles de substitution : Non

`value`

Valeur à écrire dans cette colonne de l'enregistrement de base de données.

Prend en charge les modèles de substitution : Oui

`roleArn`

L'Amazon Resource Name (ARN) du rôle qui accorde à AWS IoT l'autorisation d'écrire dans la table de la base de données Timestream. Pour de plus amples informations, veuillez consulter Prérequis.

Prend en charge les modèles de substitution : Non

`tableName`

Nom de la table de la base de données dans laquelle les enregistrements de la mesure doivent être inscrits. Voir aussi **`databaseName`**.

Supporte les modèles de substitution : API et AWS CLI uniquement

`timestamp`

Valeur à utiliser pour l'horodatage de l'entrée. Si le champ est vide, l'heure à laquelle l'entrée a été traitée est utilisée.

`unit`

Précision de la valeur d'horodatage résultant de l'expression décrite à la section `value`.

Valeurs valides: `SECONDS` | `MILLISECONDS` | `MICROSECONDS` | `NANOSECONDS` L'argument par défaut est `MILLISECONDS`.

`value`

Expression qui renvoie une valeur temporelle de longue époque.

Vous pouvez utiliser cette the section called “time_to_epoch (Chaîne, Chaîne)” fonction pour créer un horodatage valide à partir d'une valeur de date ou d'heure transmise dans la charge utile du message.

Contenu d'un enregistrement Timestream

Les données écrites dans la table Amazon Timestream par cette action incluent un horodatage, les métadonnées de l'action de règle Timestream et le résultat de l'instruction de requête de la règle.

Pour chaque attribut (mesure) du résultat de l'instruction de requête, cette action de règle écrit un enregistrement dans la table Timestream spécifiée avec ces colonnes.

Nom de la colonne	Type d’attribut	Valeur	Commentaires
dimension-name	DIMENSION	La valeur spécifiée dans l'entrée d'action de la règle Timestream.	Chaque dimension spécifiée dans l'entrée d'action de règle crée une colonne dans la base de données Timestream avec le nom de la dimension.
nom_mesure	NOM_MESURE	Le nom de l'attribut	Nom de l'attribut figurant dans le résultat de l'instruction de requête dont la valeur est spécifiée dans la `measure_value::` *data-type* colonne.
valeur_mesure : *data-type*	VALEUR_MESURE	La valeur de l'attribut dans le résultat de l'instruction de requête. Le nom de l'attribut figure dans la `measure_name` colonne.	La valeur est interprétée* et définie comme la correspondance la plus appropriée entre :`bigint`, `booleandouble`, ou. `varchar` Amazon

Nom de la colonne	Type d'attribut	Valeur	Commentaires
			Timestream crée une colonne distincte pour chaque type de données. La valeur du message peut être convertie en un autre type de données à l'aide de la `cast()` fonction contenue dans l'instruction de requête de la règle.
time	TIMESTAMP	Date et heure de l'enregistrement dans la base de données.	Cette valeur est attribuée par le moteur de règles ou par la `timestamp` propriété, si elle est définie.

* La valeur d'attribut lue à partir de la charge utile du message est interprétée comme suit. Voir le the section called "Exemples" pour une illustration de chacun de ces cas.

- Une valeur sans guillemets de `true` ou `false` est interprétée comme un `boolean` type.
- Un nombre décimal est interprété comme un `double` type.
- Une valeur numérique sans point décimal est interprétée comme un `bigint` type.
- Une chaîne entre guillemets est interprétée comme un `varchar` type.
- Les objets et les valeurs des tableaux sont convertis en chaînes JSON et stockés sous forme `varchar` de type.

Exemples

L'exemple JSON suivant définit une action de règle Timestream avec un modèle de substitution dans une AWS IoT règle.

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'iot/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "timestream": {
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_timestream",
          "tableName": "devices_metrics",
          "dimensions": [
            {
              "name": "device_id",
              "value": "${clientId()}"
            },
            {
              "name": "device_firmware_sku",
              "value": "My Static Metadata"
            }
          ],
          "databaseName": "record_devices"
        }
      }
    ]
  }
}
```

L'utilisation de l'action de règle de rubrique Timestream définie dans l'exemple précédent avec la charge utile des messages suivante entraîne l'écriture des enregistrements Amazon Timestream dans le tableau ci-dessous.

```
{
  "boolean_value": true,
  "integer_value": 123456789012,
  "double_value": 123.456789012,
  "string_value": "String value",
  "boolean_value_as_string": "true",
  "integer_value_as_string": "123456789012",
  "double_value_as_string": "123.456789012",
  "array_of_integers": [23,36,56,72],
  "array of strings": ["red", "green","blue"],
  "complex_value": {
```

```
    "simple_element": 42,
    "array_of_integers": [23,36,56,72],
    "array of strings": ["red", "green","blue"]
  }
}
```

Le tableau suivant affiche les colonnes et les enregistrements de base de données créés à l'aide de l'action de règle de rubrique spécifiée pour traiter la charge utile du message précédent. Les `device_id` colonnes `device_firmware_sku` et sont les DIMENSIONS définies dans l'action de règle du sujet. L'action de règle de sujet Timestream crée la `time` colonne et les `measure_value::*` colonnes `measure_name` et, qu'elle remplit avec les valeurs issues du résultat de l'instruction de requête de l'action de règle de sujet.

device_firmware_sku	device_id	nom_mesure	value_mesure : :bigint	valeur_mesure : varchar	valeur_mesure : double	value_mesure : :booléen	time
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE78	valeur_complexe	-	{"simple_element » : « array_of_integers » : [23,36,56,72], « tableau de chaînes » : ["rouge », « vert », « bleu"]}	-	-	26-08-2020- 22:42:16.423000000
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE78	valeur_entière en tant que chaîne	-	123456789012	-	-	26-08-2020- 22:42:16.423000000
Mes métadonné	console IoT	valeur_booléenne	-	-	-	TRUE	26-08-2020- 22:42:16.

device_firmware_sku	device_id	nom_mesure	value_mesure : :bigint	valeur_mesure : varchar	valeur_mesure : double	value_mesure : :boolé	time
es statiques	159 -0 EXAMPLE7 8						423000000
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE7 8	valeur_entière	123456789 012	-	-	-	26-08-202 0- 22:42:16. 423000000
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE7 8	valeur_chaîne	-	Valeur de chaîne	-	-	26-08-202 0- 22:42:16. 423000000
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE7 8	tableau d'entiers	-	[23,36,56 ,72]	-	-	26-08-202 0- 22:42:16. 423000000
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE7 8	tableau de chaînes	-	["rouge », « vert », « bleu"]	-	-	26-08-202 0- 22:42:16. 423000000
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE7 8	valeur_booléenne en tant que chaîne	-	TRUE	-	-	26-08-202 0- 22:42:16. 423000000

device_firmware_sku	device_id	nom_mesure	value_measure : :bigint	valeur_measure : varchar	valeur_measure : double	value_measure : :boolé	time
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE78	valeur_double	-	-	123,456789012	-	26-08-2020- 22:42:16.423000000
Mes métadonnées statiques	console IoT 159 -0 EXAMPLE78	double_value_as_string	-	123,45679	-	-	26-08-2020- 22:42:16.423000000

Résolution des problèmes d'une règle

Si vous rencontrez un problème avec vos règles, nous vous recommandons d'activer CloudWatch Logs. Vous pouvez analyser vos journaux pour déterminer s'il s'agit d'un problème d'autorisation ou si, par exemple, une condition de la clause WHERE n'a pas été respectée. Pour plus d'informations, consultez la section Configuration CloudWatch des journaux.

Accès aux ressources entre comptes à l'aide AWS IoT de règles

Vous pouvez configurer des AWS IoT règles d'accès entre comptes afin que les données ingérées sur les sujets MQTT d'un compte puissent être acheminées vers les AWS services, tels qu'Amazon SQS et Lambda, d'un autre compte. Ce qui suit explique comment configurer des AWS IoT règles pour l'ingestion de données entre comptes, qu'il s'agisse d'un sujet MQTT dans un compte ou d'une destination dans un autre compte.

Les règles entre comptes peuvent être configurées à l'aide d'autorisations basées sur les ressources sur la ressource de destination. Par conséquent, seules les destinations qui prennent en charge les autorisations basées sur les ressources peuvent être activées pour l'accès entre comptes avec des règles. AWS IoT Les destinations prises en charge incluent Amazon SQS, Amazon SNS, Amazon S3 et. AWS Lambda

> **Note**
>
> Pour les destinations prises en charge, à l'exception d'Amazon SQS, vous devez définir la règle de la même manière Région AWS que la ressource d'un autre service afin que l'action de la règle puisse interagir avec cette ressource. Pour plus d'informations sur les actions des AWS IoT règles, consultez la section Actions des AWS IoT règles. Pour plus d'informations sur l'action SQS de la règle, consultez???.

Prérequis

- Connaissance des règles AWS IoT
- Compréhension des utilisateurs, des rôles et des autorisations basées sur les ressources IAM
- Après avoir AWS CLIinstallé

Configuration entre comptes pour Amazon SQS

Scénario : le compte A envoie les données d'un message MQTT à la file d'attente Amazon SQS du compte B.

Compte AWS	Compte désigné comme	Description
1111-1111-1111	Compte A	Action de la règle `sqs:SendMessage`
2222-2222-2222	Compte B	File d'attente Amazon SQS • ARN : *arn:aws:sqs:region:2222-2222-2222:ExampleQueue* • URL : *https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue*

> **Note**
>
> Votre file d'attente Amazon SQS de destination ne doit pas nécessairement être identique à votre Région AWSAWS IoT règle. Pour plus d'informations sur l'action SQS de la règle, consultez???.

Faire la tâches du compte A

> **Remarque**
>
> Pour exécuter les commandes suivantes, votre utilisateur IAM doit être autorisé `iot:CreateTopicRule` à utiliser le nom de ressource Amazon (ARN) de la règle comme ressource et à `iam:PassRole` agir avec une ressource comme ARN du rôle.

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte A.
2. Créez un rôle IAM qui fait confiance au moteur de AWS IoT règles et qui associe une politique autorisant l'accès à la file d'attente Amazon SQS du compte B. Consultez des exemples de commandes et de documents de politique dans la section Octroi AWS IoT de l'accès requis.
3. Pour créer une règle attachée à une rubrique, exécutez la create-topic-rule commande.

   ```
   aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
   ```

   Voici un exemple de fichier de données utiles avec une règle qui insère tous les messages envoyés au `iot/test`sujet dans la file d'attente Amazon SQS spécifiée. L'instruction SQL filtre les messages et le rôle ARN AWS IoT accorde des permissions d'ajouter le message à la file d'attente Amazon SQS.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "sqs": {
       "queueUrl": "https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue",
   ```

```
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role",
    "useBase64": false
   }
  }
 ]
}
```

Pour plus d'informations sur la définition d'une action Amazon SQS dans une AWS IoT règle, consultez Actions de AWS IoT règles - Amazon SQS.

Faire la tâches du compte B

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte B.
2. Pour autoriser le compte A à accéder à la ressource de file d'attente Amazon SQS, exécutez la commande add permission.

```
aws sqs add-permission --queue-url https://sqs.region.amazonaws.com/2222-2222-2222/
ExampleQueue --label SendMessagesToMyQueue --aws-account-ids 1111-1111-1111 --
actions SendMessage
```

Configuration de comptes croisés pour Amazon SNS

Scénario : Le compte A envoie des données d'un message MQTT à une rubrique Amazon SNS du compte B.

Compte AWS	Compte désigné comme	Description
1111-1111-1111	Compte A	Action de la règle `sns:Publish`
2222-2222-2222	Compte B	ARN de rubrique Amazon SNS : *arn:aws:sns:region:2222-2222-2222:ExampleTopic*

Faire la tâches du compte A

> **Remarques**
>
> Pour exécuter les commandes suivantes, votre utilisateur IAM doit être autorisé à utiliser `iot:CreateTopicRule` la règle ARN comme ressource et à effectuer l'`iam:PassRole`action avec une ressource comme ARN de rôle.

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte A.
2. Créez un rôle IAM qui fait confiance au moteur de AWS IoT règles et qui associe une politique autorisant l'accès à la rubrique Amazon SNS du compte B. Pour des exemples de commandes et de documents de politique, consultez la section Octroi AWS IoT de l'accès requis.
3. Pour créer une règle attachée à une rubrique, exécutez la create-topic-rule commande.

   ```
   aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
   ```

   Voici un exemple de fichier de données utiles avec une règle qui insère tous les messages envoyés au `iot/test` sujet dans le sujet Amazon SNS spécifié. L'instruction SQL filtre les messages et le rôle ARN accorde les AWS IoT autorisations nécessaires pour envoyer le message au sujet Amazon SNS.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "sns": {
       "targetArn": "arn:aws:sns:region:2222-2222-2222:ExampleTopic",
       "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
      }
     }
    ]
   }
   ```

   Pour plus d'informations sur la définition d'une action Amazon SNS dans une AWS IoT règle, consultez Actions de AWS IoT règle - Amazon SNS.

Faire la tâches du compte B

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte B.
2. Pour autoriser le compte A à accéder à la ressource thématique Amazon SNS, exécutez la commande add permission.

```
aws sns add-permission --topic-arn arn:aws:sns:region:2222-2222-2222:ExampleTopic
 --label Publish-Permission --aws-account-id 1111-1111-1111 --action-name Publish
```

Configuration entre comptes pour Amazon S3

Scénario : Le compte A envoie des données d'un message MQTT vers un compartiment Amazon S3 du compte B.

Compte AWS	Compte désigné comme	Description
1111-1111-1111	Compte A	Action de la règle `s3:PutObject`
2222-2222-2222	Compte B	Compartiments Amazon S3 : *arn:aws:s3:::amzn-s3-demo-bucket*

Faire la tâches du compte A

> **Remarque**
>
> Pour exécuter les commandes suivantes, votre utilisateur IAM doit être autorisé à `iot:CreateTopicRule` utiliser la règle ARN comme ressource et à `iam:PassRole` agir avec une ressource comme ARN de rôle.

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte A.
2. Créez un rôle IAM qui fait confiance au moteur de AWS IoT règles et qui associe une politique autorisant l'accès au compartiment Amazon S3 du compte B. Pour des exemples de commandes et de documents de politique, consultez la section Octroi AWS IoT de l'accès requis.

3. Pour créer une règle attachée à votre compartiment S3 cible, exécutez la create-topic-rule commande.

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

Voici un exemple de fichier de données utiles avec une règle qui insère tous les messages envoyés au `iot/test` sujet dans le compartiment Amazon S3 spécifié. L'instruction SQL filtre les messages et le rôle ARN accorde les AWS IoT autorisations nécessaires pour ajouter le message au compartiment Amazon S3.

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "s3": {
    "bucketName": "amzn-s3-demo-bucket",
    "key": "${topic()}/${timestamp()}",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
   }
  }
 ]
}
```

Pour plus d'informations sur la façon de définir une action Amazon S3 dans une AWS IoT règle, consultez Actions de AWS IoT règle - Amazon S3.

Faire la tâches du compte B

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte B.
2. Créez une politique de compartiment qui fait confiance au principal du compte A.

   Voici un exemple de fichier de charge utile qui définit une politique de compartiment qui fait confiance au principal d'un autre compte.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AddCannedAcl",
   "Effect": "Allow",
   "Principal": {
    "AWS": [
     "arn:aws:iam::1111-1111-1111:root"
    ]
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
  }
 ]
}
```

Pour plus d'informations, veuillez consulter bucket policy examples.

3. Pour associer la politique de compartiment au compartiment spécifié, exécutez la put-bucket-policy commande.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://./amzn-s3-demo-bucket-policy.json
```

4. Pour que l'accès entre comptes fonctionne, assurez-vous que les paramètres Bloquer tout accès public sont corrects. Pour plus d'informations, consultez Security Best Practices for Amazon S3.

Configuration de comptes multiples pour AWS Lambda

Scénario : le compte A invoque une AWS Lambda fonction du compte B en transmettant un message MQTT.

Compte AWS	Compte désigné comme	Description
1111-1111-1111	Compte A	Action de la règle `lambda:InvokeFunction`
2222-2222-2222	Compte B	ARN de fonction Lambda : *arn:aws:lambda:region:2222-2222-2222:function:example-function*

Faire la tâches du compte A

> **Remarques**
>
> Pour exécuter les commandes suivantes, votre utilisateur IAM doit être autorisé à `iot:CreateTopicRule` utiliser la règle ARN comme ressource et à `iam:PassRole` agir avec la ressource comme ARN de rôle.

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte A.
2. Exécutez la create-topic-rule commande pour créer une règle qui définit l'accès entre comptes à la fonction Lambda du compte B.

   ```
   aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
   ```

   Voici un exemple de fichier de charge utile avec une règle qui insère tous les messages envoyés au `iot/test` sujet dans la fonction Lambda spécifiée. L'instruction SQL filtre les messages et l'ARN du rôle accorde AWS IoT l'autorisation de transmettre les données à la fonction Lambda.

   ```
   {
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
     {
      "lambda": {
       "functionArn": "arn:aws:lambda:region:2222-2222-2222:function:example-function"
   ```

```
    }
   }
  ]
}
```

Pour plus d'informations sur la définition d'une AWS Lambda action dans une AWS IoT règle, consultez la section Actions de AWS IoT règles - Lambda.

Faire la tâches du compte B

1. Configurez AWS CLI à l'aide de l'utilisateur IAM du compte B.
2. Exécutez la commande add permission de Lambda pour autoriser les AWS IoT règles à activer la fonction Lambda. Pour exécuter la commande suivante, votre utilisateur IAM doit être autorisé à `lambda:AddPermission` agir.

```
aws lambda add-permission --function-name example-function --region us-east-1 --
principal iot.amazonaws.com --source-arn arn:aws:iot:region:1111-1111-1111:rule/
example-rule --source-account 1111-1111-1111 --statement-id "unique_id" --action
 "lambda:InvokeFunction"
```

Options :

--principal

Ce champ autorise AWS IoT (représenté par`iot.amazonaws.com`) à appeler la fonction Lambda.

--source-arn

Ce champ confirme que cette fonction Lambda ne peut être activée que `arn:aws:iot:region:1111-1111-1111:rule/example-rule` dans les AWS IoT déclencheurs et qu'aucune autre règle du même compte ou d'un compte différent ne peut activer cette fonction Lambda.

--source-account

Ce champ confirme que cette AWS IoT fonction Lambda est activée uniquement au nom du `1111-1111-1111` compte.

> **Remarques**
>
> Si le message d'erreur « La règle est introuvable » s'affiche sur la console de votre AWS Lambda fonction sous Configuration, ignorez le message d'erreur et testez la connexion.

Gestion des erreurs (action d'erreur)

Lorsqu'il AWS IoT reçoit un message d'un appareil, le moteur de règles vérifie si le message correspond à une règle. Si c'est le cas, l'instruction de requête de la règle est évaluée et les actions de la règle sont activées, en transmettant le résultat de l'instruction de requête.

Si un problème survient lors de l'activation d'une action, le moteur de règles active une action d'erreur, si une telle action est spécifiée pour la règle. Cela peut se produire dans les cas suivants :

- Une règle n'a pas l'autorisation d'accéder à un compartiment Amazon S3.
- Une erreur de l'utilisateur entraîne un dépassement du DynamoDB provisioned throughput

> **Note**
>
> La gestion des erreurs abordée dans cette rubrique concerne les actions relatives aux règles. Pour résoudre les problèmes SQL, y compris les fonctions externes, vous pouvez configurer la AWS IoT journalisation. Pour de plus amples informations, veuillez consulter ???.

Format du message d'action d'erreur

Un seul message est généré par la règle et par message. Par exemple, si deux actions de règle échouent dans une même règle, l'action d'erreur reçoit un message contenant les deux erreurs.

Par exemple, le message d'erreur ressemble à ce qui suit :

```
{
  "ruleName": "TestAction",
  "topic": "testme/action",
  "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
  "clientId": "iotconsole-1511213971966-0",
```

```
  "base64OriginalPayload":
 "ewogICJtZXNzYWdlIjogIkhlbGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
  "failures": [
    {
      "failedAction": "S3Action",
      "failedResource": "us-east-1-s3-verify-user",
      "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error
 Code: NoSuchBucket; Request ID: 9DF5416B9B47B9AF; S3 Extended Request ID:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
 Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
  ]
}
```

ruleName

Nom de la règle qui a déclenché l'action d'erreur.

topic

Rubrique dans laquelle le message d'origine a été reçu.

cloudwatchTraceId

Une identité unique faisant référence à l'erreur se connecte CloudWatch.

clientId

ID client de l'éditeur du message.

base 64 OriginalPayload

Charge utile du message d'origine codée en base64.

échecs

failedAction

Nom de l'action qui a échoué, par exemple « S3Action ».

failedResource

Nom de la ressource. Par exemple, le nom d'un compartiment S3.

errorMessage

Description et explication de l'erreur.

Exemple d'action d'erreur

Voici un exemple de règle à laquelle a été ajoutée une action d'erreur. La règle suivante comporte une action qui écrit les données du message dans une table DynamoDB et une action d'erreur qui écrit les données dans un panier Amazon S3 :

```
{
    "sql" : "SELECT * FROM ..."
    "actions" : [{
        "dynamoDB" : {
            "table" : "PoorlyConfiguredTable",
            "hashKeyField" : "AConstantString",
            "hashKeyValue" : "AHashKey"}}
    ],
    "errorAction" : {
        "s3" : {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
            "bucketName" : "message-processing-errors",
            "key" : "${replace(topic(), '/', '-') + '-' + timestamp() + '-' +
 newuuid()}"
        }
    }
}
```

Vous pouvez utiliser n'importe quelle fonction ou modèle de substitution dans l'instruction SQL d'une action d'erreur, y compris les fonctions externes : `aws_lambda()`, `get_dynamodb()`, `get_thing_shadow()`, `get_secret()`, `machinelearning_predict()`, et `decode()`. Si une action d'erreur nécessite l'appel d'une fonction externe, l'invocation de l'action d'erreur peut entraîner une facture supplémentaire pour la fonction externe.

Les fonctions externes suivantes sont facturées de manière équivalente à celle d'une action de règle : `aws_lambda`, `get_dynamodb()`, et `get_thing_shadow()`. La `decode()`fonction ne vous est également facturée que lorsque vous décodez un message Protobuf en JSON. Pour plus de détails, consultez la page de AWS IoT Core tarification.

Pour plus d'informations sur les règles et sur la manière de spécifier une action d'erreur, consultez la section Création d'une AWS IoT règle.

Pour plus d'informations sur l'utilisation CloudWatch pour surveiller le succès ou l'échec des règles, consultezAWS IoT métriques et dimensions.

Réduction des coûts de messagerie avec Basic Ingest

Vous pouvez utiliser Basic Ingest pour envoyer en toute sécurité les données de l'appareil aux personnes Services AWS prises en charge parAWS IoT actions liées aux règles, sans frais de messagerie. Basic Ingest optimise le flux de données en supprimant le courtier de publish/subscribe messages du chemin d'ingestion.

Basic Ingest peut envoyer des messages depuis vos appareils ou applications. Les messages ont des noms de rubrique qui commencent par `$aws/rules/`*`rule_name`* les trois premiers niveaux, où *`rule_name`* est le nom de la AWS IoT règle que vous souhaitez invoquer.

Vous pouvez utiliser une règle existante avec Basic Ingest en ajoutant simplement le préfixe Basic Ingest (`$aws/rules/`*`rule_name`*) à la rubrique du message par lequel vous invoquez normalement la règle. Par exemple, si vous avez une règle nommée `BuildingManager` qui est invoquée par des messages avec des sujets tels que `Buildings/Building5/Floor2/Room201/Lights` (`"sql": "SELECT * FROM 'Buildings/#'"`), vous pouvez invoquer la même règle avec Basic Ingest en envoyant un message avec le sujet `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`.

> **Note**
>
> - Vos appareils et vos règles ne peuvent pas s'abonner aux rubriques réservées Basic Ingest. Par exemple, les AWS IoT Device Defender `num-messages-received` métriques ne sont pas émises car elles ne permettent pas de s'abonner à des sujets. Pour de plus amples informations, veuillez consulter Rubriques réservées.
> - Si vous avez besoin d'un publish/subscribe courtier pour distribuer des messages à plusieurs abonnés (par exemple, pour envoyer des messages à d'autres appareils et au moteur de règles), vous devez continuer à utiliser le courtier de AWS IoT messages pour gérer la distribution des messages. Cependant, il suffit de publier vos messages dans des rubriques autres que les rubriques Basic Ingest.

Utilisation de Basic Ingest

Avant d'utiliser Basic Ingest, vérifiez que votre appareil ou application utilise une politique sur laquelle des autorisations de publication sont activées `$aws/rules/*`. Vous pouvez également spécifier des autorisations pour des règles individuelles `$aws/rules/rule_name/*` dans la politique. Sinon, vos appareils et applications peuvent continuer à utiliser leurs connexions existantes avec AWS IoT Core.

Lorsque le message atteint le moteur de règles, il n'existe pas de différence dans l'exécution ou la gestion des erreurs entre les règles invoquées à partir de Basic Ingest et celles invoquées via des abonnements d'agent de messages.

Vous pouvez créer des règles à utiliser avec Basic Ingest. Gardez à l'esprit les points suivants :

- Le préfixe initial d'une rubrique Basic Ingest (`$aws/rules/rule_name`) n'est pas disponible pour la fonction topic(Decimal).
- Si vous définissez une règle qui est invoquée uniquement avec Basic Ingest, la clause `FROM` est facultative dans le champ `sql` de la définition `rule`. Elle est encore requise si la règle est également invoquée par d'autres messages qui doivent être envoyés via l'agent de messages (par exemple, parce que ces autres messages doivent être distribués à plusieurs abonnés). Pour de plus amples informations, veuillez consulter AWS IoT Référence SQL.
- Les trois premiers niveaux de la rubrique Basic Ingest (`$aws/rules/rule_name`) ne sont pas comptabilisés dans la limite de longueur des 8 segments ou des 256 caractères maximum pour une rubrique. Sinon, les mêmes restrictions que celles documentées dans AWS IoT Limites s'appliquent.
- Si un message contenant une rubrique Basic Ingest spécifiant une règle inactive ou inexistante est reçu, un journal des erreurs est créé dans un CloudWatch journal Amazon pour vous aider à effectuer le débogage. Pour de plus amples informations, veuillez consulter Entrées de journal du moteur de règles. Une métrique `RuleNotFound` est indiquée et vous pouvez créer des alarmes sur cette métrique. Pour de plus amples informations, veuillez consulter Métriques dans Métriques de règle.
- Vous pouvez toujours publier avec QoS 1 dans des rubriques Basic Ingest. Vous recevez un PUBACK une fois que le message a été transmis avec succès au moteur de règles. La réception d'un PUBACK ne signifie pas que vos actions de règle se sont terminées avec succès. Vous pouvez configurer une action d'erreur pour gérer les erreurs lors de l'exécution de l'action. Pour de plus amples informations, veuillez consulter Gestion des erreurs (action d'erreur).

AWS IoT Référence SQL

Dans AWS IoT, les règles sont définies à l'aide d'une syntaxe de type SQL. Les instructions SQL se composent de trois types de clauses :

SELECT

(Obligatoire) Extrait des informations de la charge utile d'un message entrant et effectue des transformations sur ces informations. Les messages à utiliser sont identifiés par le filtre de rubrique spécifié dans la clause FROM.

La clause SELECT prend en charge Types de données, Opérateurs, Fonctions, Littéraux, Instructions Case, Extensions JSON, Modèles de substitution, Requêtes d'objets imbriqués, et Charges utiles binaires.

FROM

Le filtre de sujet des messages MQTT qui identifie les messages dont les données doivent être extraites. La règle est activée pour chaque message envoyé à une rubrique MQTT qui correspond au filtre de rubrique défini ici. Requis pour les règles qui sont activées par les messages qui passent par l'agent de messages. Facultatif pour les règles qui sont activées uniquement à l'aide de la fonctionnalité Basic Ingest.

WHERE

(Facultatif) Ajoute une logique conditionnelle qui détermine si les actions spécifiées par une règle sont exécutées.

La clause WHERE prend en charge Types de données, Opérateurs, Fonctions, Littéraux, Instructions Case, Extensions JSON, Modèles de substitution et Requêtes d'objets imbriqués.

Un exemple d'instruction SQL se présente sous la forme suivante :

```
SELECT color AS rgb FROM 'topic/subtopic' WHERE temperature > 50
```

Un exemple de message MQTT (également appelé charge entrante) se présente sous la forme suivante :

```
{
    "color":"red",
    "temperature":100
```

```
}
```

Si ce message est publié dans la rubrique `'topic/subtopic'`, la règle est déclenchée et l'instruction SQL est évaluée. L'instruction SQL extrait la valeur de la propriété `color` si la propriété `"temperature"` est supérieure à 50. La clause WHERE spécifie la condition `temperature > 50`. Le mot-clé `AS` change le nom de la propriété `"color"` en `"rgb"`. Le résultat (également appelé charge sortante) se présente sous la forme suivante :

```
{
    "rgb":"red"
}
```

Ces données sont alors transférées vers l'action de la règle qui envoie les données pour traitement supplémentaire. Pour plus d'informations sur les actions de règle, consultez AWS IoT actions liées aux règles.

> **Note**
>
> Les commentaires ne sont actuellement pas pris en charge dans la syntaxe AWS IoT SQL. Les noms d'attributs contenant des espaces ne peuvent pas être utilisés comme noms de champs dans l'instruction SQL. Bien que la charge utile entrante puisse comporter des noms d'attributs contenant des espaces, ces noms ne peuvent pas être utilisés dans l'instruction SQL. Ils seront toutefois transmis à la charge utile sortante si vous utilisez une spécification de nom de champ générique (*).

Clause SELECT

La clause AWS IoT SELECT est essentiellement identique à la clause ANSI SQL SELECT, avec quelques différences mineures.

La clause SELECT prend en charge Types de données, Opérateurs, Fonctions, Littéraux, Instructions Case, Extensions JSON, Modèles de substitution, Requêtes d'objets imbriqués, et Charges utiles binaires.

Vous pouvez utiliser la clause SELECT pour extraire les informations des messages entrants MQTT. Une clause `SELECT *` peut être utilisée pour récupérer toute la charge utile du message entrant. Par exemple :

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT * FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50}
```

Si la charge utile est un objet JSON, vous pouvez référencer des clés dans l'objet. Votre charge utile sortante contient la paire clé-valeur. Par exemple :

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'topic/subtopic'
Outgoing payload: {"color":"red"}
```

Vous pouvez utiliser le même mot-clé AS pour renommer des clés. Par exemple :

```
Incoming payload published on topic 'topic/subtopic':{"color":"red", "temperature":50}
SQL:SELECT color AS my_color FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red"}
```

Vous pouvez sélectionner plusieurs éléments en les séparant par une virgule. Par exemple :

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT color as my_color, temperature as fahrenheit FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red","fahrenheit":50}
```

Vous pouvez sélectionner plusieurs éléments comprenant « * » pour ajouter des éléments dans la charge utile entrante. Par exemple :

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT *, 15 as speed FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50, "speed":15}
```

Vous pouvez utiliser le mot-clé `"VALUE"` pour produire les charges utiles sortantes qui ne sont pas des objets JSON. Avec la version SQL `2015-10-08`, vous ne pouvez sélectionner qu'un seul élément. Avec la version SQL `2016-03-23` ou une version ultérieure, vous pouvez également sélectionner un tableau à afficher en tant qu'objet de niveau supérieur.

Example

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
```

```
SQL: SELECT VALUE color FROM 'topic/subtopic'
Outgoing payload: "red"
```

Vous pouvez utiliser une syntaxe `'.'` pour explorer des objets JSON imbriqués dans la charge utile entrante. Par exemple :

```
Incoming payload published on topic 'topic/subtopic': {"color":
{"red":255,"green":0,"blue":0}, "temperature":50}
SQL: SELECT color.red as red_value FROM 'topic/subtopic'
Outgoing payload: {"red_value":255}
```

Pour plus d'informations sur l'utilisation des noms d'objets et de propriétés JSON qui incluent des caractères réservés, tels que des chiffres ou le trait d'union (moins), veuillez consulter Extensions JSON

Vous pouvez utiliser des fonctions (voir Fonctions) pour transformer la charge utile entrante. Vous pouvez utiliser des parenthèses pour le regroupement. Par exemple :

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT (temperature - 32) * 5 / 9 AS celsius, upper(color) as my_color FROM
 'topic/subtopic'
Outgoing payload: {"celsius":10,"my_color":"RED"}
```

Clause FROM

La clause FROM abonne votre règle à une rubrique ou à un filtre de rubriques. Mettez le sujet ou le filtre du sujet entre guillemets simples ('). La règle est déclenchée pour chaque message envoyé à une rubrique MQTT qui correspond au filtre de rubrique défini ici. Vous pouvez vous abonner à un groupe de sujets similaires à l'aide d'un filtre de sujet.

Exemple :

Charge utile entrante publiée dans une rubrique `'topic/subtopic'` : `{temperature: 50}`

Charge utile entrante publiée dans une rubrique `'topic/subtopic-2'` : `{temperature: 50}`

SQL : `"SELECT temperature AS t FROM 'topic/subtopic'"`.

`'topic/subtopic'` est abonné à la règle, de sorte que la charge utile entrante soit transmise à la règle. La charge utile sortante, transmise aux actions de règle, est : `{t: 50}`. La règle n'est pas

abonnée à `'topic/subtopic-2'`, donc la règle n'est pas déclenchée pour le message publié sur `'topic/subtopic-2'`.

Exemple de caractère générique # :

Vous pouvez utiliser le caractère générique « # » (multiniveau) pour correspondre à un ou plusieurs éléments de chemin particuliers :

Charge utile entrante publiée dans une rubrique `'topic/subtopic'` : `{temperature: 50}`.

Charge utile entrante publiée dans une rubrique `'topic/subtopic-2'` : `{temperature: 60}`.

Charge utile entrante publiée dans une rubrique `'topic/subtopic-3/details'` : `{temperature: 70}`.

Charge utile entrante publiée dans une rubrique `'topic-2/subtopic-x'` : `{temperature: 80}`.

SQL : `"SELECT temperature AS t FROM 'topic/#'"`.

La règle est abonnée à n'importe quel sujet commençant par. Elle est donc exécutée trois fois`'topic'`, envoyant des charges utiles sortantes de `{t: 50}` (pour le sujet/sous-sujet), (pour le sujet/sous-thème 2) et `{t: 60}` (pour) à ses actions. `{t: 70}` topic/subtopic-3/details Il n'y a pas d'abonnement à `'topic-2/subtopic-x'`, la règle n'est donc pas déclenchée pour le message `{temperature: 80}`.

Exemple de caractère générique + :

Vous pouvez utiliser le caractère générique « + » (un seul niveau) pour correspondre à tout élément de chemin particulier :

Charge utile entrante publiée dans une rubrique `'topic/subtopic'` : `{temperature: 50}`.

Charge utile entrante publiée dans une rubrique `'topic/subtopic-2'` : `{temperature: 60}`.

Charge utile entrante publiée dans une rubrique `'topic/subtopic-3/details'` : `{temperature: 70}`.

Charge utile entrante publiée dans une rubrique `'topic-2/subtopic-x'` : `{temperature: 80}`.

SQL : `"SELECT temperature AS t FROM 'topic/+'"`.

La règle est abonnée à toutes les rubriques avec deux éléments de chemin où le premier élément est `'topic'`. La règle est exécutée pour les messages envoyés à `'topic/subtopic'` e t`'topic/`

`subtopic-2'`, mais pas `'topic/subtopic-3/details'` (elle comporte plus de niveaux que le filtre de rubrique) ou `'topic-2/subtopic-x'` (elle ne commence pas par `topic`).

Clause WHERE

La clause WHERE détermine si les actions spécifiées par une règle sont exécutées. Si la clause WHERE évalue sur true, les actions de règle sont exécutées. Sinon, les actions de règle ne sont pas exécutées.

La clause WHERE prend en charge Types de données, Opérateurs, Fonctions, Littéraux, Instructions Case, Extensions JSON, Modèles de substitution et Requêtes d'objets imbriqués.

Exemple :

Charge utile entrante publiée dans `topic/subtopic` : `{"color":"red", "temperature":40}`.

SQL : `SELECT color AS my_color FROM 'topic/subtopic' WHERE temperature > 50 AND color <> 'red'`.

Dans ce cas, la règle sera déclenchée mais les actions spécifiées par cette règle ne seront pas exécutées. Il n'y aura aucune charge utile sortante.

Vous pouvez utiliser des fonctions et des opérateurs dans une clause WHERE. Toutefois, vous ne pouvez pas faire référence à des alias créés avec le mot-clé AS dans la clause SELECT. (La clause WHERE est évaluée en premier pour déterminer si la clause SELECT doit être évaluée.)

Exemple avec une charge utile non JSON :

Charge utile non JSON entrante publiée sur « rubrique/sous-rubrique » : « 80 »

SQL : `` `SELECT decode(encode(*, 'base64'), 'base64') AS value FROM 'topic/subtopic' WHERE decode(encode(*, 'base64'), 'base64') > 50 ``

Dans ce cas, la règle sera déclenchée mais les actions spécifiées par cette règle ne seront pas exécutées. La charge utile sortante sera transformée par la clause SELECT en charge utile JSON `{"value":80}`.

Types de données

Le moteur de AWS IoT règles prend en charge tous les types de données JSON.

Types de données pris en charge

Type	Signification
`Int`	Un `Int` discret. 34 chiffres maximum.
`Decimal`	Une valeur `Decimal` avec une précision de 34 chiffres, avec une magnitude non nulle de 1E-999 et une magnitude maximale de 9.999… E999. **Note** Certaines fonctions renvoient des valeurs `Decimal` avec double précision plutôt qu'avec une précision de 34 chiffres. Avec SQL V2 (2016-03-23), les valeurs numériques qui sont des nombres entiers, telles que `10.0`, sont traitées comme une valeur `Int` (`10`) au lieu de la valeur `Decimal` attendue (`10.0`). Pour traiter de manière fiable des valeurs numériques entières en tant que `Decimal` valeurs, utilisez SQL V1 (2015-10-08) pour l'instruction de requête de règle.
`Boolean`	`True` ou `False`.
`String`	Une chaîne UTF-8.
`Array`	Une série de valeurs qui ne sont pas nécessair ement du même type.
`Object`	Une valeur JSON composée d'une clé et d'une valeur. Les clés doivent être des chaînes. Les valeurs peuvent être de n'importe quel type.

Type	Signification
`Null`	`Null` comme défini par JSON. C'est une valeur réelle qui représente l'absence d'une valeur. Vous pouvez créer une valeur `Null` en utilisant le mot-clé `Null` dans votre instruction SQL. Par exemple : `"SELECT NULL AS n FROM 'topic/subtopic'"`

`Undefined`	Ce n'est pas une valeur. Non représenté dans JSON, sauf en omettant la valeur. Par exemple, dans l'objet `{"foo": null}`, la clé « foo » renvoie NULL, mais la clé « bar » renvoie `Undefined` . En interne, le langage SQL traite `Undefined` comme une valeur, mais il ne peut pas être représenté dans JSON, donc quand il est sérialisé au format JSON, les résultats sont `Undefined` .
	`{"foo":null, "bar":undefined}`
	est sérialisé au format JSON comme suit :
	`{"foo":null}`
	De même, `Undefined` est converti en chaîne vide lorsqu'il est sérialisé par lui-même. Les fonctions appelées avec des arguments non valides (par exemple, des types incorrects, un nombre d'arguments incorrect, etc.) renvoient `Undefined` .

Conversions

Le tableau suivant répertorie les résultats lorsqu'une valeur d'un type est convertie dans un autre type (lorsqu'une valeur d'un type incorrect est transmise à une fonction). Par exemple, si la fonction de

valeur absolue « abs » (qui prévoit une valeur `Int` ou `Decimal`) reçoit une valeur `String`, elle tente de convertir la valeur `String` en `Decimal`, en respectant ces règles. Dans ce cas, « abs("-5.123") » est traité comme « abs(-5.123) ».

> **Note**
>
> Il n'y a aucune tentative de conversion en `Array`, `Object`, `Null` ou `Undefined`.

En valeur décimale

Type d'argument	Résultat
`Int`	Un chiffre `Decimal` sans virgule décimale.
`Decimal`	La valeur source.
`Boolean`	`Undefined` . (Vous pouvez explicitement utiliser la fonction cast pour transformer true = 1.0, false = 0.0.)
`String`	Le moteur SQL essaie d'analyser la chaîne en tant `Decimal` que. AWS IoT tente d'analyse r les chaînes correspondant à l'expression régulière :`^-?\d+(\.\d+)?((?i)E-?\d+)?$` . « 0 », « -1.2 », « 5E-12 » sont des exemples de chaînes qui sont automatiq uement converties en valeurs `Decimal`.
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Null`.
Non défini	`Undefined` .

En Entier

Type d'argument	Résultat
`Int`	La valeur source.
`Decimal`	La valeur source arrondie à la valeur `Int` la plus proche.
`Boolean`	`Undefined` . (Vous pouvez explicitement utiliser la fonction cast pour transformer true = 1.0, false = 0.0.)
`String`	Le moteur SQL essaie d'analyser la chaîne en tant `Decimal` que. AWS IoT tente d'analyse r les chaînes correspondant à l'expression régulière :`^-?\d+(\.\d+)?((?i)E-?\d +)?$` . « 0 », « -1,2 », « 5E-12 » sont tous des exemples de chaînes converties automatiq uement en `Decimal` s. AWS IoT tente de convertir le en a`Decimal`, puis en tronque les décimales `String` pour en faire un. `Decimal` `Int`
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Null`.
Non défini	`Undefined` .

En valeur booléenne

Type d'argument	Résultat
`Int`	`Undefined` . (Vous pouvez explicitement utiliser la fonction `cast` pour transformer 0 = False, any_nonzero_value = True.)

Type d’argument	Résultat
`Decimal`	`Undefined` . (Vous pouvez explicitement utiliser la fonction cast pour transformer 0 = False, any_nonzero_value = True.)
`Boolean`	La valeur d'origine.
`String`	« true »=True et « false »=False (insensible à la casse). Les autres valeurs de chaînes sont : `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

En chaîne

Type d’argument	Résultat
`Int`	Une représentation de chaîne de la valeur `Int` en notation standard.
`Decimal`	Une chaîne représentant la valeur `Decimal`, probablement en notation scientifique.
`Boolean`	« true » ou « false ». Tout en minuscules.
`String`	La valeur d'origine.
Tableau	Le `Array` sérialisé au format JSON. La chaîne résultante est une liste de valeurs séparées par des virgules, délimitées par des crochets. Une valeur `String` est indiquée entre guillemets.

Type d'argument	Résultat
	Pas les valeurs `Decimal`, `Int`, `Boolean` et `Null`.
Objet	L'objet sérialisé au format JSON. La chaîne résultante est une liste de paires clé-valeur séparées par des virgules, qui commence et se termine par des accolades. Une valeur `String` est indiquée entre guillemets. Pas les valeurs `Decimal`, `Int`, `Boolean` et `Null`.
Null	`Undefined` .
Non défini	Non défini.

Opérateurs

Les opérateurs suivants peuvent être utilisés dans les clauses SELECT et WHERE.

Opérateur AND

Il renvoie une valeur `Boolean`. Il effectue une opération AND logique. Il renvoie la valeur true si les opérandes gauche et droit sont vrais. Sinon, il renvoie la valeur « false ». Des opérandes `Boolean` ou de chaînes « true » ou « false » sensibles à la casse sont requis.

Syntaxe : *`expression`* `AND` *`expression`*.

Opérateur AND

Opérande gauche	Opérande droit	Sortie
`Boolean`	`Boolean`	`Boolean`. Vrai si les deux opérandes sont vrais. Sinon, la valeur renvoyée est Faux.
`String`/`Boolean`	`String`/`Boolean`	Si toutes les chaînes sont « true » ou « false » (insensib les à la casse), elles sont converties en valeurs `Boolean`

Opérande gauche	Opérande droit	Sortie
		et traitées normalement en tant que valeurs *boolean* AND *boolean*.
Autre valeur	Autre valeur	Undefined .

Opérateur OR

Il renvoie une valeur Boolean. Il effectue une opération OR logique. Il renvoie la valeur true si les opérandes gauche ou droit sont vrais. Sinon, il renvoie la valeur « false ». Des opérandes Boolean ou de chaînes « true » ou « false » sensibles à la casse sont requis.

Syntaxe : *expression* OR *expression*.

Opérateur OR

Opérande gauche	Opérande droit	Sortie
Boolean	Boolean	Boolean. Vrai si l'un des deux opérandes est vrai. Sinon, la valeur renvoyée est Faux. Sinon, la valeur renvoyée est Faux.
String/Boolean	String/Boolean	Si toutes les chaînes sont « true » ou « false » (insensib les à la casse), elles sont converties en valeurs booléennes et traitées normalement en tant que valeurs *boolean* OR *boolean*.
Autre valeur	Autre valeur	Undefined .

Opérateur NOT

Il renvoie une valeur Boolean. Il effectue une opération NOT logique. Il renvoie true si l'opérande est faux. Sinon, la valeur renvoyée est true. Un opérande Boolean ou un opérande de chaîne « true » ou « false » insensible à la casse est requis.

Syntaxe : NOT *expression*.

Opérateur NOT

Opérande	Sortie
`Boolean`	`Boolean`. Vrai si l'opérande est Faux. Sinon, la valeur renvoyée est Vrai.
`String`	Si une chaîne est « true » ou « false » (insensible à la casse), elle est convertie dans la valeur booléenne correspondante, et la valeur opposée est renvoyée.
Autre valeur	`Undefined` .

Opérateur IN

Il renvoie une valeur `Boolean`. Vous pouvez utiliser l'opérateur IN dans une clause WHERE pour vérifier si une valeur correspond à une valeur d'un tableau. Elle renvoie vrai si la correspondance est trouvée, et faux dans le cas contraire.

Syntaxe : *`expression`* `IN` *`expression`*.

Opérateur IN

Opérande gauche	Opérande droit	Sortie
`Int/Decimal/String/A`	`Array`	Vrai si l'`Object`élément `Integer` `DecimalString/Array///`se trouve dans le tableau. Sinon, la valeur renvoyée est Faux.

Exemple :

```
SQL: "select * from 'a/b' where 3 in arr"

JSON: {"arr":[1, 2, 3, "three", 5.7, null]}
```

Dans cet exemple, la clause condition `where 3 in arr` sera évaluée à true car 3 est présent dans le tableau nomméarr. Par conséquent, dans l'instruction SQL, `select * from 'a/b'` sera exécuté. Cet exemple montre également que le tableau peut être hétérogène.

opérateur EXISTS

Il renvoie une valeur `Boolean`. Vous pouvez utiliser l'opérateur EXISTS dans une clause conditionnelle pour tester l'existence d'éléments dans une sous-requête. Elle renvoie true si la sous-requête renvoie un ou plusieurs éléments et false si la sous-requête ne renvoie aucun élément.

Syntaxe : *`expression`*.

Exemple :

```
SQL: "select * from 'a/b' where exists (select * from arr as a where a = 3)"

JSON: {"arr":[1, 2, 3]}
```

Dans cet exemple, la clause condition `where exists (select * from arr as a where a = 3)` sera évaluée à true car 3 est présent dans le tableau nomméarr. Par conséquent, dans l'instruction SQL, `select * from 'a/b'` sera exécuté.

Exemple :

```
SQL: select * from 'a/b' where exists (select * from e as e where foo = 2)

JSON: {"foo":4,"bar":5,"e":[{"foo":1},{"foo":2}]}
```

Dans cet exemple, la clause de condition `where exists (select * from e as e where foo = 2)` sera évaluée à true car le tableau e de l'objet JSON contient l'objet`{"foo":2}`. Par conséquent, dans l'instruction SQL, `select * from 'a/b'` sera exécuté.

> opérateur

Il renvoie une valeur `Boolean`. Il renvoie la valeur true si l'opérande gauche est supérieur à l'opérande droit. Les deux opérandes sont convertis en valeur `Decimal`, puis comparés.

Syntaxe : *`expression`* `>` *`expression`*.

> opérateur

Opérande gauche	Opérande droit	Sortie
`Int/Decimal`	`Int/Decimal`	`Boolean`. Vrai si l'opérande gauche est supérieur à l'opérande droit. Sinon, la valeur renvoyée est Faux.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes peuvent être converties en valeurs `Decimal`, puis en valeurs `Boolean`. Il renvoie la valeur true si l'opérande gauche est supérieur à l'opérande droit. Sinon, la valeur renvoyée est Faux.
Autre valeur	`Undefined .`	`Undefined .`

>= opérateur

Il renvoie une valeur `Boolean`. Il renvoie la valeur true si l'opérande gauche est supérieur ou égal à l'opérande droit. Les deux opérandes sont convertis en valeur `Decimal`, puis comparés.

Syntaxe : *`expression`* `>=` *`expression`*.

>= opérateur

Opérande gauche	Opérande droit	Sortie
`Int/Decimal`	`Int/Decimal`	`Boolean`.Vrai si l'opérande gauche est supérieur ou égal à l'opérande droit. Sinon, la valeur renvoyée est Faux.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes peuvent être converties en valeurs `Decimal`, puis en valeurs `Boolean`. Il renvoie la valeur true si l'opérande gauche est supérieur ou égal à l'opérande droit. Sinon, la valeur renvoyée est Faux.
Autre valeur	`Undefined .`	`Undefined .`

< opérateur

Il renvoie une valeur `Boolean`. Il renvoie la valeur true si l'opérande gauche est inférieur à l'opérande droit. Les deux opérandes sont convertis en valeur `Decimal`, puis comparés.

Syntaxe : *`expression`* `<` *`expression`*.

< opérateur

Opérande gauche	Opérande droit	Sortie
`Int/Decimal`	`Int/Decimal`	`Boolean`. Vrai si l'opérande gauche est inférieur à l'opérande droit. Sinon, la valeur renvoyée est Faux.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes peuvent être converties en valeurs `Decimal`, puis en valeurs `Boolean`. Il renvoie la valeur true si l'opérande gauche est inférieur à l'opérande droit. Sinon, la valeur renvoyée est Faux.
Autre valeur	`Undefined`	`Undefined`

<= opérateur

Il renvoie une valeur `Boolean`. Il renvoie la valeur true si l'opérande gauche est inférieur ou égal à l'opérande droit. Les deux opérandes sont convertis en valeur `Decimal`, puis comparés.

Syntaxe: *`expression`* `<=` *`expression`*.

<= opérateur

Opérande gauche	Opérande droit	Sortie
`Int/Decimal`	`Int/Decimal`	`Boolean`. Vrai si l'opérande gauche est inférieur ou égal à l'opérande droit. Sinon, la valeur renvoyée est Faux.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes peuvent être converties en valeurs `Decimal`, puis en valeurs `Boolean`. Il renvoie la valeur

Opérande gauche	Opérande droit	Sortie
		true si l'opérande gauche est inférieur ou égal à l'opérande droit. Sinon, la valeur renvoyée est Faux.
Autre valeur	`Undefined`	`Undefined`

<> opérateur

Il renvoie une valeur `Boolean`. Il renvoie la valeur true si les opérandes gauche et droit ne sont pas égaux. Sinon, la valeur renvoyée est false.

Syntaxe : *`expression`* <> *`expression`*.

<> opérateur

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	Vrai si l'opérande gauche est différent de l'opérande droit. Sinon, la valeur renvoyée est Faux.
`Decimal`	`Decimal`	Vrai si l'opérande gauche est différent de l'opérande droit. Sinon, la valeur renvoyée est Faux. `Int` est convertie en valeur `Decimal` avant d'être comparée.
`String`	`String`	Vrai si l'opérande gauche est différent de l'opérande droit. Sinon, la valeur renvoyée est Faux.
Tableau	Tableau	Vrai si les éléments de chaque opérande sont différents et ne sont pas dans le même ordre. Sinon, la valeur renvoyée est Faux
Objet	Objet	Vrai si les clés et les valeurs de chaque opérande sont différentes. Sinon, la valeur renvoyée est Faux. L'ordre de keys/values n'a pas d'importance.
Null	Null	Faux.

Opérande gauche	Opérande droit	Sortie
N'importe quelle valeur	`Undefined`	Non défini.
`Undefined`	N'importe quelle valeur	Non défini.
Type non compatible	Type non compatible	Vrai.

= opérateur

Il renvoie une valeur `Boolean`. Il renvoie la valeur true si les opérandes gauche et droit sont égaux. Sinon, la valeur renvoyée est false.

Syntaxe : *`expression`* = *`expression`*.

= opérateur

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	Vrai si l'opérande gauche est identique à l'opérande droit. Sinon, la valeur renvoyée est Faux.
`Decimal`	`Decimal`	Vrai si l'opérande gauche est identique à l'opérande droit. Sinon, la valeur renvoyée est Faux. `Int` est convertie en valeur `Decimal` avant d'être comparée.
`String`	`String`	Vrai si l'opérande gauche est identique à l'opérande droit. Sinon, la valeur renvoyée est Faux.
Tableau	Tableau	Vrai si les éléments de chaque opérande sont égaux et sont dans le même ordre. Sinon, la valeur renvoyée est Faux.

Opérande gauche	Opérande droit	Sortie
Objet	Objet	Vrai si les clés et valeurs de chaque opérande sont identiques. Sinon, la valeur renvoyée est Faux. L'ordre de keys/values n'a pas d'importance.
N'importe quelle valeur	`Undefined`	`Undefined` .
`Undefined`	N'importe quelle valeur	`Undefined` .
Type non compatible	Type non compatible	Faux.

+ opérateur

Le signe « + » est un opérateur surchargé. Il peut être utilisé pour l'addition ou la concaténation de chaines.

Syntaxe : *`expression`* + *`expression`*.

+ opérateur

Opérande gauche	Opérande droit	Sortie
`String`	N'importe quelle valeur	Il convertit l'opérande droit en chaîne et l'ajoute à la fin de l'opérande gauche.
N'importe quelle valeur	`String`	Il convertit l'opérande gauche en chaîne et ajoute l'opérande droit à la fin de l'opérande gauche converti.
`Int`	`Int`	Valeur `Int`. Il ajoute les opérandes ensemble.
`Int/Decimal`	`Int/Decimal`	Valeur `Decimal`. Il ajoute les opérandes ensemble.
Autre valeur	Autre valeur	`Undefined` .

- opérateur

Il soustrait l'opérande droit de l'opérande gauche.

Syntaxe : *expression* `-` *expression*.

- opérateur

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	Valeur `Int`. Il soustrait l'opérande droit de l'opérande gauche.
`Int/Decimal`	`Int/Decimal`	Valeur `Decimal`. Il soustrait l'opérande droit de l'opérande gauche.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes sont correctement converties en valeurs décimales, une valeur `Decimal` est renvoyée. Il soustrait l'opérande droit de l'opérande gauche. Sinon, la valeur renvoyée est `Undefined` .
Autre valeur	Autre valeur	`Undefined` .
Autre valeur	Autre valeur	`Undefined` .

* opérateur

Il multiplie l'opérande gauche par l'opérande droit.

Syntaxe : *expression* `*` *expression*.

* opérateur

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	Valeur `Int`. Il multiplie l'opérande gauche par l'opérande droit.

Opérande gauche	Opérande droit	Sortie
`Int/Decimal`	`Int/Decimal`	Valeur `Decimal`. Il multiplie l'opérande gauche par l'opérande droit.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes sont correctement converties en valeurs décimales, une valeur `Decimal` est renvoyée. Il multiplie l'opérande gauche par l'opérande droit. Sinon, la valeur renvoyée est `Undefined` .
Autre valeur	Autre valeur	`Undefined` .

/ opérateur

Il divise l'opérande gauche par l'opérande droit.

Syntaxe : *`expression`* `/` *`expression`*.

/ opérateur

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	Valeur `Int`. Il divise l'opérande gauche par l'opérande droit.
`Int/Decimal`	`Int/Decimal`	Valeur `Decimal`. Il divise l'opérande gauche par l'opérande droit.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes sont correctement converties en valeurs décimales, une valeur `Decimal` est renvoyée. Il divise l'opérande gauche par l'opérande droit. Sinon, la valeur renvoyée est `Undefined` .
Autre valeur	Autre valeur	`Undefined` .

% opérateur

Il renvoie le reste résultant de la division de l'opérande gauche par l'opérande droit.

Syntaxe : *`expression`* `%` *`expression`*.

% opérateur

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	Valeur `Int`. Il renvoie le reste résultant de la division de l'opérande gauche par l'opérande droit.
`String/Int/Deci`	`String/Int/Deci`	Si toutes les chaînes sont correctement converties en valeurs décimales, une valeur `Decimal` est renvoyée. Il renvoie le reste résultant de la division de l'opérande gauche par l'opérande droit. Sinon la valeur est renvoy, `Undefined` .
Autre valeur	Autre valeur	`Undefined` .

Fonctions

Vous pouvez utiliser les fonctions intégrées suivantes dans les clauses SELECT ou WHERE de vos expressions SQL.

abs(Decimal)

Il renvoie la valeur absolue d'un nombre. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Par exemple, `abs(-5)` renvoie 5.

Type d'argument	Résultat
`Int`	`Int`, la valeur absolue de l'argument.
`Decimal`	`Decimal`, la valeur absolue de l'argument.

Type d'argument	Résultat
`Boolean`	`Undefined` .
`String`	`Decimal`. Le résultat est la valeur absolue de l'argument. Si la chaîne ne peut être pas convertie, le résultat est `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

accountid()

Renvoie l'ID du compte qui possède la règle comme une valeur `String`. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`accountid()` = "123456789012"

acos(Decimal)

Renvoie le cosinus inverse d'un nombre en radians. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `acos(0)` = 1,5707963267948966

Type d'argument	Résultat
`Int`	`Decimal` (avec double précision), le cosinus inverse de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .

Type d’argument	Résultat
`Decimal`	`Decimal` (avec double précision), le cosinus inverse de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal`, le cosinus inverse de l'argument. Si la chaîne ne peut être pas convertie, le résultat est `Undefined` . Des résultats imaginaires sont retournés sous la forme `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

asin(Decimal)

Renvoie le sinus inverse d'un nombre en radians. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `asin(0)` = 0,0

Type d’argument	Résultat
`Int`	`Decimal` (avec double précision), le sinus inverse de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .

Type d’argument	Résultat
`Decimal`	`Decimal` (avec double précision), le sinus inverse de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), le sinus inverse de l'argument. Si la chaîne ne peut être pas convertie, le résultat est `Undefined` . Des résultats imaginaires sont retournés sous la forme `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

atan(Decimal)

Renvoie la tangente inverse d'un nombre en radians. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `atan(0)` = 0,0

Type d’argument	Résultat
`Int`	`Decimal` (avec double précision), la tangente inverse de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .

Type d'argument	Résultat
`Decimal`	`Decimal` (avec double précision), la tangente inverse de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal`, la tangente inverse de l'argumen t. Si la chaîne ne peut être pas convertie, le résultat est `Undefined` . Des résultats imaginaires sont retournés sous la forme `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

atan2(Decimal, Decimal)

Il renvoie l'angle en radians, entre l'axe des X positifs et le point (x, y) défini dans les deux arguments. L'angle est positif pour les angles sans le sens contraire des aiguilles d'une montre (moitié supérieure du plan, y > 0) et négatif pour les angles dans le sens des aiguilles d'une montre (moitié inférieure du plan, y < 0). Les arguments `Decimal` sont arrondis pour une meilleure précision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `atan2(1, 0)` = 1,5707963267948966

Type d'argument	Type d'argument	Résultat
`Int/Decimal`	`Int/Decimal`	`Decimal` (avec double précisio entre l'axe des X et le point (x, y

Type d'argument	Type d'argument	Résultat
`Int/Decimal/String`	`Int/Decimal/String`	`Decimal`, la tangente inverse d décrit. Si une chaîne ne peut pa convertie, le résultat est `Undef`
Autre valeur	Autre valeur	`Undefined` .

aws_lambda(functionArn, inputJson)

Appelle la fonction Lambda spécifiée en transmettant le paramètre `inputJson` à la fonction Lambda et renvoie les données JSON générées par la fonction Lambda.

Arguments

Argument	Description
`functionArn`	ARN de la fonction Lambda à appeler. La fonction Lambda doit renvoyer des données JSON.
`inputJson`	Données JSON en entrée transmises à la fonction Lambda. Pour transmettre des requêtes d'objets imbriqués et des littéraux, vous devez utiliser la version SQL 2016-03-23.

Vous devez accorder AWS IoT `lambda:InvokeFunction` des autorisations pour appeler la fonction Lambda spécifiée. L'exemple suivant montre comment accorder l'autorisation `lambda:InvokeFunction` à l'aide de l' AWS CLI :

```
aws lambda add-permission --function-name "function_name"
--region "region"
--principal iot.amazonaws.com
--source-arn arn:aws:iot:us-east-1:account_id:rule/rule_name
--source-account "account_id"
--statement-id "unique_id"
--action "lambda:InvokeFunction"
```

Les arguments de la commande add-permission sont les suivants :

--function-name

Nom de la fonction Lambda. Vous ajoutez une nouvelle autorisation pour mettre à jour la politique de ressources de la fonction.

--région

Celui Région AWS de votre compte.

--principal

Mandataire qui obtient l'autorisation. Cela devrait être `iot.amazonaws.com` pour AWS IoT autoriser l'appel d'une fonction Lambda.

--source-arn

ARN de la règle. Vous pouvez utiliser la get-topic-rule AWS CLI commande pour obtenir l'ARN d'une règle.

--source-account

L' Compte AWS endroit où la règle est définie.

--statement-id

Identifiant unique de l'instruction.

--action

L’action Lambda que vous souhaitez autoriser dans cette déclaration. Pour autoriser AWS IoT à invoquer une fonction,Lambda spécifiez `lambda:InvokeFunction`.

> **Important**
>
> Si vous ajoutez une autorisation pour un AWS IoT principal sans fournir le `source-arn` ou`source-account`, toute autorisation Compte AWS qui crée une règle avec votre action Lambda peut déclencher des règles à partir desquelles appeler votre fonction Lambda. AWS IoT Pour plus d’informations, veuillez consulter Modèle d’autorisation Lambda.

Soit une charge utile de message JSON comme suit :

```
{
    "attribute1": 21,
    "attribute2": "value"
```

```
}
```

La fonction `aws_lambda` peut être utilisée pour appeler la fonction Lambda, comme suit :

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
 {"payload":attribute1}) as output FROM 'topic-filter'
```

Si vous souhaitez transmettre l'intégralité de la charge utile des messages MQTT, vous pouvez spécifier la charge utile JSON en utilisant « * », comme dans l'exemple suivant.

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) as output
 FROM 'topic-filter'
```

`payload.inner.element` sélectionne les données à partir des messages publiés dans la rubrique « rubrique/sous-rubrique ».

`some.value` sélectionne les données à partir de la sortie générée par la fonction Lambda.

> **Note**
>
> Le moteur de règles limite la durée d'exécution des fonctions Lambda. Les appels de fonction Lambda provenant de règles doivent être terminés en moins de 2 000 millisecondes.

bitand(Int, Int)

Il effectue une opération AND au niveau du bit sur des représentations binaires des deux arguments `Int`(-convertis). Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `bitand(13, 5)` = 5

Type d'argument	Type d'argument	Résultat
`Int`	`Int`	`Int`, une opération AND au nive des deux arguments.
`Int/Decimal`	`Int/Decimal`	`Int`, une opération AND au nive des deux arguments. Tous les r

Type d'argument	Type d'argument	Résultat
		non-`Int` sont arrondis à la valeu inférieure la plus proche. Si l'un arguments ne peut pas être con valeur `Int`, le résultat `Undefin`
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, une opération AND au nive des deux arguments. Toutes les sont converties en valeurs décir arrondies à la valeur `Int` inférie proche. Si la conversion échoue est `Undefined` .
Autre valeur	Autre valeur	`Undefined` .

bitor(Int, Int)

Il effectue une opération OR au niveau du bit des représentations binaires des deux arguments. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `bitor(8, 5)` = 13

Type d'argument	Type d'argument	Résultat
`Int`	`Int`	`Int`, une opération OR au nivea deux arguments.
`Int/Decimal`	`Int/Decimal`	`Int`, une opération OR au nivea des deux arguments. Tous les non-`Int` sont arrondis à la valeu inférieure la plus proche. Si la c échoue, le résultat est `Undefin`
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, une opération OR au nivea les deux arguments. Toutes les sont converties en valeurs décir arrondies à la valeur `Int` inférie

Type d'argument	Type d'argument	Résultat
		proche. Si la conversion échoue est `Undefined` .
Autre valeur	Autre valeur	`Undefined` .

bitxor(Int, Int)

Il effectue une opération XOR au niveau du bit sur des représentations binaires des deux arguments `Int`(-convertis). Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `bitor(13, 5)` = 8

Type d'argument	Type d'argument	Résultat
`Int`	`Int`	`Int`, une opération XOR au nive sur les deux arguments.
`Int/Decimal`	`Int/Decimal`	`Int`, une opération XOR au nive bit sur les deux arguments. Les non-`Int` sont arrondis à la valeu inférieure la plus proche.
`Int/Decimal/String`	`Int/Decimal/String`	`Int`, un XOR au niveau du bit s arguments. Les chaînes sont co en décimales et arrondies à la v inférieure la plus proche. Si une échoue, le résultat est `Undefin`
Autre valeur	Autre valeur	`Undefined` .

bitnot(Int)

Il effectue une opération NOT au niveau du bit sur des représentations binaires de l'argument `Int`(-converti). Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `bitnot(13)` = 2

Type d'argument	Résultat
`Int`	`Int`, une opération NOT au niveau du bit de l'argument.
`Decimal`	`Int`, une opération NOT au niveau du bit de l'argument. La valeur `Decimal` est arrondie à la valeur `Int` inférieure la plus proche.
`String`	`Int`, une opération NOT au niveau du bit de l'argument. Les chaînes sont convertie s en valeurs décimales et arrondies à la valeur `Int` inférieure la plus proche. Si une conversion échoue, le résultat est `Undefined` .
Autre valeur	Autre valeur.

cast()

Convertit une valeur d'un type de données en un autre. La conversion se comporte principalement comme les conversions standard, avec en outre la capacité de convertir des chiffres vers/depuis des valeurs booléennes. Si vous AWS IoT ne pouvez pas déterminer comment convertir un type en un autre, le résultat est`Undefined`. Prise en charge par SQL 2015-10-08 et versions ultérieures. Format : fonte (*`value`*s*`type`*).

Exemple :

```
cast(true as Int) = 1
```

Les mots-clés suivants peuvent apparaître après « as » lors de l'appel de `cast` :

Pour SQL versions 2015-10-08 et 2016-03-23

Mot clé	Résultat
`String`	Il convertit une valeur en `String`.

Mot clé	Résultat
Nvarchar	Il convertit une valeur en `String`.
Texte	Il convertit une valeur en `String`.
Ntext	Il convertit une valeur en `String`.
varchar	Il convertit une valeur en `String`.
`Int`	Il convertit une valeur en `Int`.
Entier	Il convertit une valeur en `Int`.
Double	Transforme la valeur en `Decimal` (avec une double précision).

En outre, pour SQL version 2016-03-23

Mot clé	Résultat
`Decimal`	Il convertit une valeur en `Decimal`.
Booléen	Il convertit une valeur en `Boolean`.
`Boolean`	Il convertit une valeur en `Boolean`.

Règles de conversion de types :

Conversion en décimal

Type d’argument	Résultat
`Int`	Un chiffre `Decimal` sans virgule décimale.
`Decimal`	La valeur source.

Type d’argument	Résultat
	Note Avec SQL V2 (2016-03-23), les valeurs numériques qui sont des nombres entiers, telles que `10.0`, renvoient une valeur `Int` (`10`) au lieu de la valeur `Decimal` attendue (`10.0`). Pour convertir de manière fiable des valeurs numériques entières en tant que valeurs `Decimal`, utilisez SQL V1 (2015-10-08) pour l’instruction de requête de règle.
`Boolean`	true = 1.0, false = 0.0.
`String`	Tente d'analyser la chaîne en tant que `Decimal`. AWS IoT tente d'analyser les chaînes correspondant à l'expression regex : ^-?\d+(\.\d+)?((?i)E-?\d+)?$. « 0 », « -1.2 », « 5E-12 » sont des exemples de chaînes qui sont automatiquement converties en valeurs décimales.
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

Conversion en entier

Type d'argument	Résultat
`Int`	La valeur source.
`Decimal`	La valeur source arrondie à la valeur `Int` inférieure la plus proche.
`Boolean`	true = 1.0, false = 0.0.
`String`	Tente d'analyser la chaîne en tant que `Decimal`. AWS IoT tente d'analyser les chaînes correspondant à l'expression regex : ^-?\d+(\.\d+)?((?i)E-?\d+)?$. « 0 », « -1.2 », « 5E-12 » sont des exemples de chaînes qui sont automatiquement converties en valeurs décimales. AWS IoT tente de convertir la chaîne en valeur `Decimal`, puis de l'arrondir à la valeur `Int` inférieure la plus proche.
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

Conversion en valeur **`Boolean`**

Type d'argument	Résultat
`Int`	0 = False, any_nonzero_value = True.
`Decimal`	0 = False, any_nonzero_value = True.
`Boolean`	La valeur source.

Type d’argument	Résultat
`String`	« true »=True et « false »=False (insensib le à la casse). Autres valeurs de chaînes = `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

Conversion en chaîne

Type d’argument	Résultat
`Int`	Une représentation de chaîne de la valeur `Int` en notation standard.
`Decimal`	Une chaîne représentant la valeur `Decimal`, probablement en notation scientifique.
`Boolean`	« true » ou « false », tout en minuscules.
`String`	La valeur source.
Tableau	Le tableau sérialisé au format JSON. La chaîne résultante consiste en une liste séparée par des virgules et délimitée par des crochets. `String` est entre guillemets, à l'inverse de `Decimal`, `Int` et `Boolean`.
Objet	L'objet sérialisé au format JSON. La chaîne JSON est une liste de paires clé-valeur séparées par des virgules, délimitées par des accolades. `String` est entre guillemet

Type d’argument	Résultat
	s, à l'inverse de `Decimal`, `Int`, `Boolean` et `Null`.
Null	`Undefined` .
Non défini	`Undefined` .

ceil(Decimal)

Arrondit la valeur `Decimal` donnée à la valeur `Int` supérieure la plus proche. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`ceil(1.2)` = 2

`ceil(-1.2)` = -1

Type d’argument	Résultat
`Int`	`Int`, la valeur d'argument.
`Decimal`	`Int`, la valeur `Decimal` arrondie à la valeur `Int` supérieure la plus proche.
`String`	`Int`. La chaîne est convertie en valeur `Decimal` et arrondie à la valeur `Int` supérieure la plus proche. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Autre valeur	`Undefined` .

chr(String)

Renvoie le caractère ASCII qui correspond à l'argument `Int` donné. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`chr(65)` = "A".

`chr(49)` = "1".

Type d'argument	Résultat
`Int`	Le caractère correspondant à la valeur ASCII spécifiée. Si l'argument n'est pas une valeur ASCII valide, le résultat est `Undefined` .
`Decimal`	Le caractère correspondant à la valeur ASCII spécifiée. L'argument `Decimal` est arrondi à la valeur `Int` inférieure la plus proche. Si l'argument n'est pas une valeur ASCII valide, le résultat est `Undefined` .
`Boolean`	`Undefined` .
`String`	Si la valeur `String` peut être convertie en valeur `Decimal`, elle est arrondie à la valeur `Int` inférieure la plus proche. Si l'argument n'est pas une valeur ASCII valide, le résultat est `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Autre valeur	`Undefined` .

clientid()

Retourne l'ID du client MQTT en envoyant le message, ou une valeur `n/a` si le message n'a pas été pas envoyé via MQTT. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`clientid()` = "123456789012"

concat()

Concatène des tableaux ou des chaînes. Cette fonction accepte n'importe quel nombre d'arguments et renvoie une valeur `String` ou `Array`. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`concat()` = `Undefined`.

`concat(1)` = "1".

`concat([1, 2, 3], 4)` = [1, 2, 3, 4].

`concat([1, 2, 3], "hello")` = [1, 2, 3, "bonjour"]

`concat("con", "cat")` = "concat"

`concat(1, "hello")` = "bonjour1"

`concat("he","is","man")` = "heisman"

`concat([1, 2, 3], "hello", [4, 5, 6])` = [1, 2, 3, "bonjour", 4, 5, 6]

Nombre d'arguments	Résultat
0	`Undefined` .
1	L'argument est renvoyé non modifié.
2+	Si un argument est une valeur `Array`, le résultat est un seul tableau contenant l'ensemble des arguments. Si aucun argument n'est une valeur de tableau, et qu'un argument au moins est une valeur `String`, le résultat est la concaténation des représentations de `String` de tous les arguments. Des arguments sont

Nombre d'arguments	Résultat
	convertis en chaînes à l’aide des conversio ns standard précédemment répertoriées.

cos(Decimal)

Renvoie le cosinus d'un nombre en radians. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`cos(0)` = 1.

Type d’argument	Résultat
`Int`	`Decimal` (avec double précision), le cosinus de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .
`Decimal`	`Decimal` (avec double précision), le cosinus de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), le cosinus de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` . Des résultats imaginaires sont retournés sous la forme `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .

Type d'argument	Résultat
Null	`Undefined` .
Non défini	`Undefined` .

cosh(Decimal)

Renvoie le cosinus hyperbolique d'un nombre en radians. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `cosh(2.3)` = 5.037220649268761.

Type d'argument	Résultat
`Int`	`Decimal` (avec double précision), le cosinus hyperbolique de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .
`Decimal`	`Decimal` (avec double précision), le cosinus hyperbolique de l'argument. Des résultats imaginaires sont retournés sous la forme `Undefined` .
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), le cosinus hyperbolique de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` . Des résultats imaginaires sont retournés sous la forme `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .

Type d'argument	Résultat
Null	`Undefined` .
Non défini	`Undefined` .

décoder (valeur, schéma de décodage)

Utilisez la fonction `decode` pour décoder une valeur codée. Si la chaîne décodée est un document JSON, un objet adressable est renvoyé. Sinon, la chaîne décodée est renvoyée sous forme de chaîne. La fonction renvoie NULL si la chaîne ne peut pas être décodée. Cette fonction prend en charge le décodage des chaînes codées en base64 et du format de message Protocol Buffer (protobuf).

Pris en charge par SQL 2016-03-23 et versions ultérieures.

value

Une valeur de chaîne ou l'une des expressions valides, telles que définies dans AWS IoT Référence SQL, qui renvoie une chaîne.

decodingScheme

Chaîne littérale représentant le schéma utilisé pour décoder la valeur. À l'heure actuelle, uniquement `'base64'` et `'proto'` sont pris en charge.

Décodage de chaînes codées en base64

Dans cet exemple, la charge utile du message inclut une valeur codée.

```
{
    encoded_temp: "eyAidGVtcGVyYXR1cmUiOiAzMyB9Cg=="
}
```

La fonction `decode` de cette instruction SQL décode la valeur de la charge utile du message.

```
SELECT decode(encoded_temp,"base64").temperature AS temp from 'topic/subtopic'
```

Le décodage de la valeur `encoded_temp` permet d'obtenir le document JSON valide suivant, qui permet à l'instruction SELECT de lire la valeur de température.

```
{ "temperature": 33 }
```

Le résultat de l'instruction SELECT dans cet exemple est affiché ici.

```
{ "temp": 33 }
```

Si la valeur décodée n'était pas un document JSON valide, la valeur décodée serait renvoyée sous forme de chaîne.

Décodage de la charge utile des messages protobuf

Vous pouvez utiliser la fonction SQL de décodage pour configurer une règle capable de décoder la charge utile de votre message protobuf. Pour plus d'informations, veuillez consulter la section Décodage des charges utiles des messages protobuf.

> **Important**
>
> Si vous omettez `source-arn` ou `source-account` si vous définissez des autorisations pour un AWS IoT principal, n'importe qui Compte AWS peut invoquer votre fonction de décodage par le biais d'autres AWS IoT règles. Pour sécuriser votre fonction, consultez les politiques relatives aux compartiments dans le guide de l'utilisateur d'Amazon Simple Storage Service.

La signature de la fonction ressemble à ce qui suit :

```
decode(<ENCODED DATA>, 'proto', '<S3 BUCKET NAME>', '<S3 OBJECT KEY>', '<PROTO NAME>',
 '<MESSAGE TYPE>')
```

`ENCODED DATA`

Spécifie les données codées en protobuf à décoder. Si l'intégralité du message envoyé à la règle est constituée de données codées en protobuf, vous pouvez référencer la charge utile binaire entrante brute à l'aide de * Sinon, ce champ doit être une chaîne JSON codée en base-64 et une référence à la chaîne peut être transmise directement.

1) Pour décoder une charge utile entrante protobuf binaire brute :

```
decode(*, 'proto', ...)
```

2) Pour décoder un message codé en protobuf représenté par une chaîne codée en base64 « a.b » :

```
decode(a.b, 'proto', ...)
```

`proto`

Spécifie les données à décoder dans un format de message protobuf. Si vous spécifiez `base64` au lieu de `proto`, cette fonction décodera les chaînes codées en base64 au format JSON.

`S3 BUCKET NAME`

Le nom du compartiment Amazon S3 dans lequel vous avez chargé votre fichier `FileDescriptorSet`.

`S3 OBJECT KEY`

Clé d'objet qui spécifie le fichier `FileDescriptorSet` dans le compartiment Amazon S3.

`PROTO NAME`

Le nom du fichier `.proto` (à l'exception de l'extension) à partir duquel le fichier `FileDescriptorSet` a été généré.

`MESSAGE TYPE`

Nom de la structure du message protobuf dans le fichier `FileDescriptorSet`, à laquelle les données à décoder doivent être conformes.

Voici un exemple d'expression SQL utilisant la fonction SQL de décodage :

```
SELECT VALUE decode(*, 'proto', 's3-bucket', 'messageformat.desc', 'myproto',
 'messagetype') FROM 'some/topic'
```

- `*`

  Représente une charge utile binaire entrante, conforme au type de message protobuf appelé `mymessagetype`
- `messageformat.desc`

  Le fichier `FileDescriptorSet` stocké dans un compartiment Amazon S3 nommé `s3-bucket`.
- `myproto`

Le fichier `.proto` d'origine utilisé pour générer le fichier `FileDescriptorSet` nommé `myproto.proto`.

- `messagetype`

  Le type de message appelé `messagetype` (ainsi que toutes les dépendances importées) tel que défini dans `myproto.proto`.

encode(value, encodingScheme)

Utilisez la fonction `encode` pour encoder la charge utile, qui peut être constituée de données non-JSON, dans sa représentation de chaîne basée sur le schéma d'encodage. Pris en charge par SQL 2016-03-23 et versions ultérieures.

value

Une des expressions valides, telles que définies dans la AWS IoT Référence SQL. Vous pouvez spécifier * pour encoder la charge utile dans son ensemble, qu'elle soit ou non au format JSON. Si vous fournissez une expression, le résultat de l'évaluation est converti en une chaîne avant d'être codé.

encodingScheme

Chaîne littérale qui représente le schéma de codage à utiliser. Actuellement, seul `'base64'` est pris en charge.

endswith(String, String)

Renvoie une valeur `Boolean` indiquant si le premier argument `String` se termine par le deuxième argument `String`. Si l'un des arguments est `Null` ou `Undefined`, le résultat a la valeur `Undefined`. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Par exemple : `endswith("cat","at")` = true.

Type d'argument 1	Type d'argument 2	Résultat
`String`	`String`	Vrai si le premier argument se t dans le second argument. Sinor renvoyée est Faux.

Type d'argument 1	Type d'argument 2	Résultat
Autre valeur	Autre valeur	Les deux arguments sont conve chaînes à l'aide des règles de c n standard. Vrai si le premier ar termine dans le second argume la valeur renvoyée est Faux. Si arguments est `Null` ou `Undefi` résultat a la valeur `Undefined`

exp(Decimal)

Renvoie la valeur augmentée vers l'argument `Decimal`. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `exp(1)` = e.

Type d’argument	Résultat
`Int`	`Decimal` (avec double précision), argument puissance e.
`Decimal`	`Decimal` (avec double précision), argument puissance e.
`String`	`Decimal` (avec double précision), argument puissance e. Si la valeur `String` ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Autre valeur	`Undefined` .

floor(Decimal)

Arrondit la valeur `Decimal` donnée à la valeur `Int` inférieure la plus proche. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`floor(1.2)` = 1

`floor(-1.2)` = -2

Type d'argument	Résultat
`Int`	`Int`, la valeur d'argument.
`Decimal`	`Int`, la valeur `Decimal` arrondie à la valeur `Int` inférieure la plus proche.
`String`	`Int`. La chaîne est convertie en valeur `Decimal` et arrondie à la valeur `Int` inférieure la plus proche. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Autre valeur	`Undefined` .

get

Extrait une valeur à partir d'un type de collection (tableau, chaîne, objet). Aucune conversion n'est appliquée au premier argument. Une conversion s'applique comme documenté dans le tableau au deuxième argument. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`get(["a", "b", "c"], 1)` = "b"

`get({"a":"b"}, "a")` = "b"

`get("abc", 0)` = « a ».

Type d'argument 1	Type d'argument 2	Résultat
Tableau	Tout type (converti valeur `Int`)	L'élément à l'index de base zéro valeur `Array` fourni par le deux

Type d'argument 1	Type d'argument 2	Résultat
		argument (converti en `Int`). Si l n échoue, le résultat est `Undef` Si l'index est en dehors des limi valeur `Array` (négatif ou **>=** arr le résultat est `Undefined` .
Chaîne	Tout type (converti valeur `Int`)	Le caractère est à l'index de bas la chaîne fournie par le deuxièm (converti en `Int`). Si la conversi le résultat est `Undefined` . Si l est en dehors des limites de la c (négatif ou **>=** string.length), le r `Undefined` .
Objet	`String` (aucune conversion appliquée)	La valeur stockée dans le premi (l'objet) correspondant à la clé c fournie comme deuxième argum
Autre valeur	N'importe quelle valeur	`Undefined` .

get_dynamodb (TableName,,,,, partitionKeyName ROLearn) partitionKeyValue sortKeyName sortKeyValue

Récupère des données d'une table DynamoDB. `get_dynamodb()` vous permet d'interroger une table DynamoDB pendant l'évaluation d'une règle. Vous pouvez filtrer ou augmenter les charges utiles des messages à l'aide des données extraites de DynamoDB. Pris en charge par SQL 2016-03-23 et versions ultérieures.

`get_dynamodb()` accepte les paramètres suivants :

tableName

Nom de la table DynamoDB à interroger.

partitionKeyName

Nom de la clé de partition. Pour plus d'informations, veuillez consulter Clés DynamoDB.

partitionKeyValue

Valeur de la clé de partition utilisée pour identifier un enregistrement. Pour plus d'informations, veuillez consulter Clés DynamoDB.

sortKeyName

(Facultatif) Nom de la clé de tri. Ce paramètre n'est requis que si la table DynamoDB interrogée utilise une clé composite. Pour plus d'informations, veuillez consulter Clés DynamoDB.

sortKeyValue

(Facultatif) Valeur de la clé de tri. Ce paramètre n'est requis que si la table DynamoDB interrogée utilise une clé composite. Pour plus d'informations, veuillez consulter Clés DynamoDB.

roleArn

ARN d'un rôle IAM qui accorde l'accès à la table DynamoDB. Le moteur de règles assume ce rôle pour accéder à la table DynamoDB en votre nom. Évitez d'utiliser un rôle trop permissif. Accordez au rôle uniquement les autorisations requises par la règle. L'exemple de stratégie suivant accorde l'accès à une table DynamoDB.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:GetItem",
            "Resource": "arn:aws:dynamodb:aws-region:account-id:table/table-
name"
        }
    ]
}
```

À titre d'exemple d'utilisation de `get_dynamodb()`, supposons que vous disposez d'une table DynamoDB contenant l'ID d'appareil et les informations d'emplacement de tous vos appareils connectés à AWS IoT. L'instruction SELECT suivante utilise la fonction `get_dynamodb()` pour récupérer l'emplacement de l'ID d'appareil spécifié :

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/
topic'
```

> **Note**
>
> - Vous pouvez appeler `get_dynamodb()` une fois au maximum par instruction SQL. L'appel de `get_dynamodb()` plusieurs fois dans une même instruction SQL entraîne la fin de la règle sans invoquer aucune action.
> - Si `get_dynamodb()` renvoie plus de 8 Ko de données, l'action de la règle ne peut pas être invoquée.

get_mqtt_property (nom)

Fait référence à l'un MQTT5 des en-têtes suivants :`contentType`, `payLoadFormatIndicatorresponseTopic`, et`correlationData`. Cette fonction prend l'une des chaînes littérales suivantes comme argument :`content_type`, `format_indicator response_topic`, et`correlation_data`. Pour plus d'informations, veuillez consulter la table des arguments de fonction suivante.

contentType

Chaîne : codée en UTF-8 qui décrit le contenu du message de publication.

payLoadFormatIndicateur

Chaîne : une valeur de chaîne qui indique si la charge utile est formatée en UTF-8. Les valeurs valides sont `UNSPECIFIED_BYTES` et `UTF8_DATA`.

Rubrique de réponse

Chaîne : Chaîne codée en UTF-8 utilisée comme nom de rubrique pour un message de réponse. La rubrique de réponse permet de décrire la rubrique dans laquelle le récepteur doit effectuer la publication dans le cadre du flux demande-réponse. La rubrique ne doit pas contenir de caractères génériques.

Données de corrélation

Chaîne : Les données binaires codées en base64 utilisées par l'expéditeur du message de demande pour identifier la demande à laquelle le message de réponse correspond lorsqu'il est reçu.

Le tableau suivant indique les arguments de fonction acceptables et les types de retour associés pour la fonction `get_mqtt_property` :

Arguments de la fonction

SQL	Type de données renvoyé (le cas échéant)	Type de données renvoyé (s'il n'est pas présent)
`get_mqtt_property( "format_indicator")`	Chaîne (UNSPECIF IED_BYTES ou _DATA) UTF8	Chaîne (UNSPECIFIED_BYTES)
`get_mqtt_property( "content_type")`	Chaîne	Non défini
`get_mqtt_property( "response_topic")`	Chaîne	Non défini
`get_mqtt_property( "correlation_data")`	Chaîne codée en base64	Non défini
`get_mqtt_property( "some_invalid_name ")`	Non défini	Non défini

L'exemple de règles SQL suivant fait référence à l'un MQTT5 des en-têtes suivants :`contentType`, `payLoadFormatIndicatorresponseTopic`, et`correlationData`.

```
SELECT *, get_mqtt_property('content_type') as contentType,
          get_mqtt_property('format_indicator') as payloadFormatIndicator,
          get_mqtt_property('response_topic') as responseTopic,
          get_mqtt_property('correlation_data') as correlationData
FROM 'some/topic'
```

get_secret (SecreTid, SecretType, clé, roLearn)

Récupère la valeur du champ chiffré `SecretString` ou `SecretBinary` de la version actuelle d'un secret dans AWS Secrets Manager. Pour plus d'informations sur la création et la gestion de secrets CreateSecret, consultez les UpdateSecretsections, et PutSecretValue.

`get_secret()` accepte les paramètres suivants :

secretId

Chaîne : Amazon Resource Name (ARN) ou nom convivial du secret à récupérer.

Type de secret

Chaîne : type secret. Valeurs valides : `SecretString` | `SecretBinary`.

SecretString

- Pour les secrets que vous créez sous forme d'objets JSON à l'aide de APIs, de AWS CLI, ou de la AWS Secrets Manager console :
  - Si vous spécifiez une valeur pour le paramètre `key`, cette fonction renvoie la valeur de la clé spécifiée.
  - Si vous ne spécifiez pas de valeur pour le paramètre `key`, cette fonction renvoie l'objet JSON complet.
- Pour les secrets que vous créez en tant qu'objets non JSON en utilisant le APIs ou le AWS CLI :
  - Si vous spécifiez une valeur pour le paramètre `key`, cette fonction échoue avec une exception.
  - Si vous ne spécifiez pas de valeur pour le paramètre `key`, cette fonction renvoie le contenu du secret.

SecretBinary

- Si vous spécifiez une valeur pour le paramètre `key`, cette fonction échoue avec une exception.
- Si vous ne spécifiez aucune valeur du paramètre `key`, cette fonction renvoie la valeur secrète sous forme de chaîne UTF-8 codée en base64.

clé

(Facultatif) Chaîne : nom de la clé à l'intérieur d'un objet JSON stocké dans le champ `SecretString` d'un secret. Utilisez cette valeur lorsque vous souhaitez récupérer uniquement la valeur d'une clé stockée dans un secret au lieu de récupérer l'intégralité de l'objet JSON.

Si vous spécifiez une valeur pour ce paramètre et que le secret ne contient aucun objet JSON dans son champ `SecretString`, cette fonction échoue avec une exception.

roleArn

Chaîne : un ARN de rôle avec des autorisations `secretsmanager:GetSecretValue` et `secretsmanager:DescribeSecret`.

> **Note**
>
> Cette fonction renvoie toujours la version actuelle du secret (la version avec la balise `AWSCURRENT`). Le moteur de AWS IoT règles met en cache chaque secret pendant 15 minutes maximum. Par conséquent, le moteur de règles peut prendre jusqu'à 15 minutes pour mettre à jour un secret. Cela signifie que si vous récupérez un secret jusqu'à 15 minutes après une mise à jour avec AWS Secrets Manager, cette fonction peut renvoyer la version précédente.
> Cette fonction n'est pas mesurée, mais des AWS Secrets Manager frais s'appliquent. En raison du mécanisme de mise en cache secret, le moteur de règles appelle AWS Secrets Manager occasionnellement. Le moteur de règles étant un service entièrement distribué, il est possible que vous receviez plusieurs appels d'API Secrets Manager depuis le moteur de règles pendant la fenêtre de mise en cache de 15 minutes.

Exemples :

Vous pouvez utiliser la fonction `get_secret` dans un en-tête d'authentification dans le cadre d'une action de règle HTTPS, comme dans l'exemple d'authentification par clé d'API suivant.

```
"API_KEY": "${get_secret('API_KEY', 'SecretString', 'API_KEY_VALUE',
 'arn:aws:iam::12345678910:role/getsecret')}"
```

Pour plus d'informations sur l'action de règle HTTPS, veuillez consulter the section called "HTTP".

get_thing_shadow(thingName, shadowName, roleARN)

Renvoie le shadow spécifié de l'objet spécifié. Pris en charge par SQL 2016-03-23 et versions ultérieures.

thingName

Chaîne : nom de l'objet dont vous souhaitez récupérer le shadow.

shadowName

(Facultatif) Chaîne : nom du shadow. Ce paramètre est requis uniquement quand vous référencez des shadows nommés.

roleArn

Chaîne : un ARN de rôle avec une autorisation `iot:GetThingShadow`.

Exemples :

Lorsqu'elle est utilisée avec un shadow nommé, fournissez le paramètre `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","MyThingShadow","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

Lorsqu'elle est utilisée avec un shadow non nommé, omettez le paramètre `shadowName`.

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

get_user_properties () userPropertyKey

Références aux propriétés utilisateur, qui est un type d'en-tête de propriété pris en charge dans MQTT5.

UserProperty

Chaîne : une propriété utilisateur est une paire clé-valeur. Cette fonction prend la clé comme argument et renvoie un tableau de toutes les valeurs correspondant à la clé associée.

Arguments de la fonction

Pour les propriétés utilisateur suivantes dans les en-têtes des messages :

Clé	Valeur
une clé	une valeur
une clé différente	une valeur différente
une clé	valeur avec clé dupliquée

Le tableau suivant présente le comportement SQL attendu :

SQL	Type de données de retour	Valeur de données de retour
get_user_properties (« une clé »)	Tableau de chaînes	`['some value', 'value with duplicate key']`
get_user_properties (« une clé »)	Tableau de chaînes	`['a different value']`
get_user_properties ( )	Tableau d'objets de paire clé-valeur	`[{'"some key": "some value"'}, {"other key": "a different value"}, {"some key": "value with duplicate key"}]`
get_user_properties (« clé inexistante »)	Non défini	

L'exemple de règles SQL suivant fait référence aux propriétés utilisateur (un type d'en-tête de MQTT5 propriété) dans la charge utile :

```
SELECT *, get_user_properties('user defined property key') as userProperty
FROM 'some/topic'
```

Fonctions de hachage

AWS IoT fournit les fonctions de hachage suivantes :

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

Toutes les fonctions de hachage prévoit un argument de type chaîne. Le résultat est la valeur hachée de cette chaîne. Les conversions de chaîne standard s'appliquent aux arguments non-chaîne. Toutes les fonctions de hachage sont prises en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`md2("hello")` = "a9046c73e00331af68917d3804f70655"

`md5("hello")` = "5d41402abc4b2a76b9719d911017c592"

indexof(String, String)

Renvoie le premier index (de base 0) du deuxième argument comme une sous-chaîne dans le premier argument. Les deux arguments doivent être des chaînes. Les arguments qui ne sont pas des chaînes sont soumis aux règles de conversion de chaînes standard. Cette fonction ne s'applique pas aux tableaux, uniquement aux chaînes. Pris en charge par SQL 2016-03-23 et versions ultérieures.

Exemples :

`indexof("abcd", "bc")` = 1

isNull()

Retourne la valeur true si la valeur de l'argument est `Null`. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`isNull(5)` = false.

`isNull(Null)` = vrai.

Type d'argument	Résultat
`Int`	false
`Decimal`	false
`Boolean`	false
`String`	false
`Array`	false
`Object`	false
`Null`	vrai
`Undefined`	false

isUndefined()

Retourne la valeur true si l'argument est `Undefined`. Pris en charge par SQL 2016-03-23 et versions ultérieures.

Exemples :

`isUndefined(5)` = false.

`isUndefined(floor([1,2,3]))` = vrai.

Type d'argument	Résultat
`Int`	false
`Decimal`	false
`Boolean`	false
`String`	false
`Array`	false
`Object`	false
`Null`	false
`Undefined`	true

length(String)

Renvoie le nombre de caractères dans la chaîne fournie. Les règles de conversion standard s'appliquent aux arguments non-`String`. Pris en charge par SQL 2016-03-23 et versions ultérieures.

Exemples :

`length("hi")` = 2

`length(false)` = 5

ln(Decimal)

Renvoie le logarithme naturel de l'argument Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `ln(e)` = 1.

Type d'argument	Résultat
`Int`	`Decimal` (avec double précision), le logarithme naturel de l'argument.
`Decimal`	`Decimal` (avec double précision), le logarithme naturel de l'argument.
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), le logarithme naturel de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

log(Decimal)

Renvoie le logarithme 10 de base de l'argument Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `log(100)` = 2.0.

Type d'argument	Résultat
`Int`	`Decimal` (avec double précision), le logarithme de base 10 de l'argument.

Type d'argument	Résultat
`Decimal`	`Decimal` (avec double précision), le logarithme de base 10 de l'argument.
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), le logarithme de base 10 de l'argument. Si la valeur `String` ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

lower(String)

Renvoie la version en minuscules de la valeur de `String` donnée. Les arguments non-chaîne sont convertis en chaînes à l'aide des règles de conversion standard. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`lower("HELLO")` = "bonjour".

`lower(["HELLO"])` = "[\"bonjour\"]".

lpad(String, Int)

Renvoie l'argument `String`, complété à gauche par le nombre d'espaces spécifié par le deuxième argument. L'argument `Int` doit être compris entre 0 et 1000. Si la valeur fournie se situe en dehors de cette plage valide, l'argument est défini sur la valeur valide la plus proche (0 ou 1 000). Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`lpad("hello", 2) = "  hello"`.

`lpad(1, 3) = "   1"`

Type d'argument 1	Type d'argument 2	Résultat
`String`	`Int`	`String`, l'argument `String` fou complété à gauche par un nomb d'espaces égal à la valeur `Int`.
`String`	`Decimal`	L'argument `Decimal` est arrond `Int` inférieure la plus proche, et t `String` est complété à gauch nombre d'espaces spécifié.
`String`	`String`	Le deuxième argument est conv valeur `Decimal`, qui est arrondi `Int` inférieure la plus proche, et t `String` est complété à gauch nombre d'espaces spécifié. Si le argument ne peut pas être conv valeur `Int`, le résultat `Undefin`
Autre valeur	`Int/Decimal/String`	La première valeur est convertie valeur `String` à l'aide des conv ns standard, puis la fonction LP appliquée sur cette valeur `Stri` ne peut pas être convertie, le ré `Undefined` .
N'importe quelle valeur	Autre valeur	`Undefined` .

ltrim(String)

Supprime tous les espaces de début (tabulations et espaces) de la valeur `String` fournie. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`Ltrim(" h i ")` = "bonjour".

Type d'argument	Résultat
`Int`	La représentation `String` de `Int` avec tous les espaces de début supprimés.
`Decimal`	La représentation `String` de `Decimal` avec tous les espaces de début supprimés.
`Boolean`	La représentation `String` de la valeur booléenne (« true » ou « false ») avec tous les espaces de début supprimés.
`String`	L'argument avec tous les espaces de début supprimés.
Tableau	La représentation `String` de `Array` (à l'aide des règles de conversion standard) avec tous les espaces de début supprimés.
Objet	La représentation `String` de l'objet (à l'aide des règles de conversion standard) avec tous les espaces de début supprimés.
Null	`Undefined` .
Non défini	`Undefined` .

machinelearning_predict(modelId, roleArn, record)

Utilisez cette `machinelearning_predict` fonction pour faire des prédictions en utilisant les données d'un message MQTT basé sur un modèle Amazon SageMaker AI. Prise en charge par SQL 2015-10-08 et versions ultérieures. Les arguments de la fonction `machinelearning_predict` sont :

modelId

L'ID du modèle sur lequel doit être réalisée la prévision. Le point de terminaison en temps réel du modèle doit être activé.

roleArn

Le rôle IAM qui dispose d'une stratégie avec les autorisations `machinelearning:Predict` et `machinelearning:GetMLModel` permet d'accéder au modèle par rapport auquel la prévision doit être réalisée.

record

Les données à transmettre à l'API SageMaker AI Predict. Elles doivent être représentées sous la forme d'un objet JSON à couche unique. Si l'enregistrement est un objet JSON multiniveau, il est mis à plat en sérialisant ses valeurs. Par exemple, le code JSON suivant :

```
{ "key1": {"innerKey1": "value1"}, "key2": 0}
```

deviendrait :

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0}
```

La fonction renvoie un objet JSON dans les champs suivants :

predictedLabel

Classification de l'entrée basée sur le modèle.

détails

Contient les attributs suivants :

PredictiveModelType

Type de modèle. Les valeurs valides sont REGRESSION, BINARY, MULTICLASS.

Algorithm

Algorithme utilisé par l' SageMaker IA pour faire des prédictions. La valeur doit être SGD.

predictedScores

Contient le score de classification brut correspondant à chaque étiquette.

predictedValue

La valeur prédite par l' SageMaker IA.

mod(Decimal, Decimal)

Renvoie le reste résultant de la division du premier argument par le deuxième argument. Équivalent à remainder(Decimal, Decimal). Vous pouvez également utiliser « % » comme opérateur infixe pour la même fonctionnalité modulo. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `mod(8, 3)` = 2.

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	`Int`, les premier et deuxième ar pour lesquels vous voulez exécu fonctionnalité Modulo.
`Int/Decimal`	`Int/Decimal`	`Decimal`, le premier argument deuxième opérande pour lesque voulez exécuter la fonctionnalité
`String/Int/Decimal`	`String/Int/Decimal`	Si toutes les chaînes sont conve en décimales, le résultat est le p argument divisé par le deuxième Sinon la valeur est renvoy, Und
Autre valeur	Autre valeur	`Undefined` .

nanol (,) AnyValue AnyValue

Renvoie le premier argument s'il s'agit d'une valeur `Decimal` valide. Sinon, le deuxième argument est renvoyé. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `Nanvl(8, 3)` = 8.

Type d'argument 1	Type d'argument 2	Sortie
Non défini	N'importe quelle valeur	Le deuxième argument.
Null	N'importe quelle valeur	Le deuxième argument.
`Decimal` (NaN)	N'importe quelle valeur	Le deuxième argument.
`Decimal` (non-NaN)	N'importe quelle valeur	Le premier argument.
Autre valeur	N'importe quelle valeur	Le premier argument.

newuuid()

Retourne un UUID aléatoire de 16 octets. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple: `newuuid()` = `123a4567-b89c-12d3-e456-789012345000`

numbytes(String)

Renvoie le nombre d'octets dans l'encodage UTF-8 de la chaîne fournie. Les règles de conversion standard s'appliquent aux arguments non-`String`. Pris en charge par SQL 2016-03-23 et versions ultérieures.

Exemples :

`numbytes("hi")` = 2

`numbytes("€")` = 3

parse_time(String, Long[, String])

Utilisez cette `parse_time` fonction pour formater un horodatage dans un format lisible par l'homme date/time . Pris en charge par SQL 2016-03-23 et versions ultérieures. Pour convertir une chaîne d'horodatage en millisecondes, veuillez consulter time_to_epoch (Chaîne, Chaîne).

La fonction `parse_time` attend les arguments suivants :

pattern

(String) Un date/time modèle qui suit les formats Joda-Time.

timestamp

(Long) Heure à formater en millisecondes depuis l'époque Unix. Voir la fonction timestamp().

timezone

(Chaîne) Fuseau horaire de la date/heure mise en forme. La valeur par défaut est « UTC ». La fonction prend en charge les fuseaux horaires Joda-Time. Cet argument est facultatif.

Exemples :

Lorsque ce message est publié dans la rubrique « A/B », la charge utile `{"ts": "1970.01.01 AD at 21:46:40 CST"}` est envoyée au compartiment S3 :

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", 100000000,
 'America/Belize' ) as ts FROM 'A/B'",

        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

Lorsque ce message est publié dans la rubrique « A/B », une charge utile similaire à `{"ts": "2017.06.09 AD at 17:19:46 UTC"}` (mais avec la date et l'heure du moment) est envoyée au compartiment S3 :

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
```

```
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", timestamp() ) as ts
 FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

`parse_time()` peut également servir de modèle de substitution. Par exemple, lorsque ce message est publié dans la rubrique « A/B », la charge utile est envoyée au compartiment S3 avec la clé = « 2017 » :

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT * FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [{
            "s3": {
                "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                "bucketName": "BUCKET_NAME",
                "key": "${parse_time('yyyy', timestamp(), 'UTC')}"
            }
        }],
        "ruleName": "RULE_NAME"
    }
}
```

power(Decimal, Decimal)

Renvoie le premier argument augmenté vers le deuxième argument. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `power(2, 5)` = 32.0.

Type d'argument 1	Type d'argument 2	Sortie
`Int/Decimal`	`Int/Decimal`	Une valeur `Decimal` (avec dou précision), le premier argument la puissance du deuxième argu
`Int/Decimal/String`	`Int/Decimal/String`	Une valeur `Decimal` (avec dou précision), le premier argument à la puissance du deuxième arg Toutes les chaînes sont convert décimales. Si tout valeur `Strin` être convertie en `Decimal`, le r `Undefined` .
Autre valeur	Autre valeur	`Undefined` .

principal()

Renvoie le principal utilisé par le terminal pour l'authentification, en fonction de la manière dont le message déclencheur a été publié. Le tableau suivant décrit le mandataire renvoyé pour chaque méthode et protocole de publication.

Méthode de publication du message	Protocole	Type d'informations d'identificat
Client MQTT	MQTT	Certificat d'appareil X.509
AWS IoT client MQTT pour console	MQTT	Utilisateur ou rôle IAM

Méthode de publication du message	Protocole	Type d'informations d'identificat
AWS CLI	HTTP	Utilisateur ou rôle IAM
AWS IoT SDK de l'appareil	MQTT	Certificat d'appareil X.509
AWS IoT SDK de l'appareil	MQTT terminé WebSocket	Utilisateur ou rôle IAM

Les exemples suivants illustrent les différents types de valeurs qui peuvent être renvoyés par `principal()` :

- Empreinte du certificat X.509 : `ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373`
- ID de rôle IAM et nom de session : `ABCD1EFG3HIJK2LMNOP5:my-session-name`
- Renvoie un ID utilisateur : `ABCD1EFG3HIJK2LMNOP5`

rand()

Renvoie une valeur pseudo aléatoire, uniformément distribuée en double entre 0,0 et 1,0. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`rand()` = 0.8231909191640703

regexp_matches(String, String)

Renvoie la valeur true si la chaîne (le premier argument) contient un élément correspondant à l'expression régulière (le deuxième argument). Si vous l'utilisez `|` dans l'expression régulière, utilisez-la avec `()`.

Exemples :

`regexp_matches("aaaa", "a{2,}")` = vrai.

`regexp_matches("aaaa", "b")` = false.

`regexp_matches("aaa", "(aaa|bbb)")` = vrai.

`regexp_matches("bbb", "(aaa|bbb)")` = vrai.

`regexp_matches("ccc", "(aaa|bbb)")` = false.

Premier argument :

Type d'argument	Résultat
`Int`	La représentation `String` de la valeur `Int`.
`Decimal`	La représentation `String` de la valeur `Decimal`.
`Boolean`	La représentation `String` de la valeur booléenne (« vrai » ou « faux »).
`String`	La valeur `String`.
Tableau	La représentation `String` de la valeur `Array` (à l'aide des règles de conversion standard).
Objet	La représentation `String` de l'objet (à l'aide des règles de conversion standard).
Null	`Undefined` .
Non défini	`Undefined` .

Deuxième argument :

Il doit s'agir d'une expression regex valide. Les types non-chaîne sont convertis en valeurs `String` à l'aide des règles de conversion standard. Selon le type, la chaîne résultante peut ne pas être une expression régulière valide. Si l'argument (converti) n'est pas un regex valide, le résultat est `Undefined`.

regexp_replace(String, String, String)

Remplace toutes les occurrences du deuxième argument (expression régulière) figurant dans le premier argument par le troisième argument. Fait référence aux groupes de capture avec « $ ». Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`regexp_replace("abcd", "bc", "x")` = "axd".

`regexp_replace("abcd", "b(.*)d", "$1")` = "ac".

Premier argument :

Type d'argument	Résultat
`Int`	La représentation `String` de la valeur `Int`.
`Decimal`	La représentation `String` de la valeur `Decimal`.
`Boolean`	La représentation `String` de la valeur booléenne (« vrai » ou « faux »).
`String`	La valeur source.
Tableau	La représentation `String` de la valeur `Array` (à l'aide des règles de conversion standard).
Objet	La représentation `String` de l'objet (à l'aide des règles de conversion standard).
Null	`Undefined` .
Non défini	`Undefined` .

Deuxième argument :

Il doit s'agir d'une expression regex valide. Les types non-chaîne sont convertis en valeurs `String` à l'aide des règles de conversion standard. Selon le type, la chaîne résultante peut ne pas être une

expression régulière valide. Si l'argument (converti) n'est pas une expression regex valide, le résultat est `Undefined`.

Troisième argument :

Il doit s'agir d'une chaîne de remplacement regex valide. (Peut faire référence à d'autres groupes de capture.) Les types non-chaîne sont convertis en valeurs `String` à l'aide des règles de conversion standard. Si l'argument (converti) n'est pas une chaîne de remplacement regex valide, le résultat est `Undefined`.

regexp_substr(String, String)

Recherche la première correspondance du deuxième paramètre (regex) dans le premier paramètre. Fait référence aux groupes de capture avec « $ ». Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`regexp_substr("hihihello", "hi")` = "bonjour"

`regexp_substr("hihihello", "(hi)*")` = "hihi"

Premier argument :

Type d’argument	Résultat
`Int`	La représentation `String` de la valeur `Int`.
`Decimal`	La représentation `String` de la valeur `Decimal`.
`Boolean`	La représentation `String` de la valeur booléenne (« vrai » ou « faux »).
`String`	L'argument `String`.
Tableau	La représentation `String` de la valeur `Array` (à l'aide des règles de conversion standard).
Objet	La représentation `String` de l'objet (à l'aide des règles de conversion standard).

Type d'argument	Résultat
Null	`Undefined` .
Non défini	`Undefined` .

Deuxième argument :

Il doit s'agir d'une expression regex valide. Les types non-chaîne sont convertis en valeurs `String` à l'aide des règles de conversion standard. Selon le type, la chaîne résultante peut ne pas être une expression régulière valide. Si l'argument (converti) n'est pas une expression regex valide, le résultat est `Undefined`.

remainder(Decimal, Decimal)

Renvoie le reste résultant de la division du premier argument par le deuxième argument. Équivalent à mod(Decimal, Decimal). Vous pouvez également utiliser « % » comme opérateur infixe pour la même fonctionnalité modulo. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `remainder(8, 3)` = 2.

Opérande gauche	Opérande droit	Sortie
`Int`	`Int`	`Int`, les premier et deuxième ar pour lesquels vous voulez exécu fonctionnalité Modulo.
`Int/Decimal`	`Int/Decimal`	`Decimal`, le premier argument deuxième opérande pour lesque voulez exécuter la fonctionnalité
`String/Int/Decimal`	`String/Int/Decimal`	Si toutes les chaînes sont conve en décimales, le résultat est le p argument divisé par le deuxièm Sinon la valeur est renvoy, Und
Autre valeur	Autre valeur	`Undefined` .

replace(String, String, String)

Remplace toutes les occurrences du deuxième argument par le troisième argument dans le premier argument. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`replace("abcd", "bc", "x")` = `"axd"`.

`replace("abcdabcd", "b", "x")` = `"axcdaxcd"`.

Tous les arguments

Type d'argument	Résultat
`Int`	La représentation `String` de la valeur `Int`.
`Decimal`	La représentation `String` de la valeur `Decimal`.
`Boolean`	La représentation `String` de la valeur booléenne (« vrai » ou « faux »).
`String`	La valeur source.
Tableau	La représentation `String` de la valeur `Array` (à l'aide des règles de conversion standard).
Objet	La représentation `String` de l'objet (à l'aide des règles de conversion standard).
Null	`Undefined` .
Non défini	`Undefined` .

rpad(String, Int)

Renvoie l'argument chaîne, complété à droite par le nombre d'espaces spécifié dans le deuxième argument. L'argument `Int` doit être compris entre 0 et 1000. Si la valeur fournie se situe en dehors

de cette plage valide, l'argument est défini sur la valeur valide la plus proche (0 ou 1 000). Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`rpad("hello", 2)` = "`hello  `".

`rpad(1, 3)` = "`1  `".

Type d'argument 1	Type d'argument 2	Résultat
`String`	`Int`	L'argument `String` est complété à droite par un nombre d'espaces égal à la valeur `Int` fournie.
`String`	`Decimal`	L'argument `Decimal` est arrondi à la valeur `Int` inférieure la plus proche, et la chaîne est complétée à droite par un nombre d'espaces égal à la valeur `Int` fournie.
`String`	`String`	Le deuxième argument est converti en une valeur `Decimal`, qui est arrondie à la valeur `Int` inférieure la plus proche. L'argument `String` est complété à droite par un nombre d'espaces égal à la valeur `Int` fournie.
Autre valeur	`Int`/`Decimal`/`String`	La première valeur est convertie en une

Type d'argument 1	Type d'argument 2	Résultat
		valeur `String` à l'aide des conversions standard, puis la fonction RPAD est appliquée sur cette valeur `String`. Si elle ne peut pas être convertie, le résultat est `Undefined` .
N'importe quelle valeur	Autre valeur	`Undefined` .

round(Decimal)

Arrondit la valeur `Decimal` donnée à la valeur `Int` la plus proche. Si la valeur `Decimal` se situe à équidistance entre deux valeurs `Int` (par exemple, 0,5), la valeur `Decimal` est arrondie à la valeur supérieure. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `Round(1.2)` = 1.

`Round(1.5)` = 2.

`Round(1.7)` = 2.

`Round(-1.1)` = -1.

`Round(-1.5)` = -2.

Type d'argument	Résultat
`Int`	L'argument.
`Decimal`	La valeur `Decimal` est arrondie à la valeur `Int` inférieure la plus proche.
`String`	La valeur `Decimal` est arrondie à la valeur `Int` inférieure la plus proche. Si la chaîne

Type d’argument	Résultat
	ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Autre valeur	`Undefined` .

rtrim(String)

Supprime tous les espaces de fin (tabulations et espaces) de la valeur `String` fournie. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`rtrim(" h i ")` = " sa lut "

Type d’argument	Résultat
`Int`	La représentation `String` de la valeur `Int`.
`Decimal`	La représentation `String` de la valeur `Decimal`.
`Boolean`	La représentation `String` de la valeur booléenne (« vrai » ou « faux »).
Tableau	La représentation `String` de la valeur `Array` (à l'aide des règles de conversion standard).
Objet	La représentation `String` de l'objet (à l'aide des règles de conversion standard).
Null	`Undefined` .
Non défini	`Undefined`

sign(Decimal)

Renvoie le signe d'un chiffre donné. Lorsque le signe de l'argument est positif, la valeur 1 et renvoyée. Lorsque le signe de l'argument est négatif, la valeur -1 et renvoyée. Si l'argument est 0, la valeur 0 est renvoyée. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`sign(-7)` = -1.

`sign(0)` = 0.

`sign(13)` = 1.

Type d'argument	Résultat
`Int`	`Int`, le signe de la valeur `Int`.
`Decimal`	`Int`, le signe de la valeur `Decimal`.
`String`	`Int`, le signe de la valeur `Decimal`. La chaîne est convertie en une valeur `Decimal`, et le signe de la valeur `Decimal` est renvoyée. Si la valeur `String` ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` . Prise en charge par SQL 2015-10-08 et versions ultérieures.
Autre valeur	`Undefined` .

sin(Decimal)

Renvoie le sinus d'un nombre en radians. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `sin(0)` = 0,0

Type d'argument	Résultat
`Int`	`Decimal` (avec double précision), le sinus de l'argument.
`Decimal`	`Decimal` (avec double précision), le sinus de l'argument.
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), le sinus de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
`Undefined`	`Undefined` .

sinh(Decimal)

Renvoie le sinus hyperbolique d'un nombre. Les valeurs `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Le résultat est une valeur `Decimal` de double précision. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `sinh(2.3)` = 4,936961805545957

Type d'argument	Résultat
`Int`	`Decimal` (avec double précision), le sinus hyperbolique de l'argument.
`Decimal`	`Decimal` (avec double précision), le sinus hyperbolique de l'argument.

Type d'argument	Résultat
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), le sinus hyperbolique de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

sourceip()

Récupère l'adresse IP d'un appareil ou du routeur qui s'y connecte. Si votre appareil est connecté directement à Internet, la fonction renvoie l'adresse IP source de l'appareil. Si votre appareil est connecté à un routeur connecté à Internet, la fonction renvoie l'adresse IP source du routeur. Prise en charge par SQL version 23/03/2016. `sourceip()` ne prend aucun paramètre.

> **Important**
>
> L'adresse IP source publique d'un appareil est souvent l'adresse IP de la dernière passerelle de traduction d'adresses réseau (NAT), telle que le routeur ou le modem câble de votre fournisseur d'accès Internet.

Exemples :

```
sourceip()="192.158.1.38"
```

```
sourceip()="1.102.103.104"
```

```
sourceip()="2001:db8:ff00::12ab:34cd"
```

Exemple SQL :

```
SELECT *, sourceip() as deviceIp FROM 'some/topic'
```

Exemples d'utilisation de la fonction sourceip () dans les actions de AWS IoT Core règles :

Exemple 1

L'exemple suivant montre comment appeler la fonction () en tant que modèle de substitution dans une action DynamoDB.

```
{
 "topicRulePayload": {
  "sql": "SELECT * AS message FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "dynamoDB": {
     "tableName": "my_ddb_table",
     "hashKeyField": "key",
     "hashKeyValue": "${sourceip()}",
     "rangeKeyField": "timestamp",
     "rangeKeyValue": "${timestamp()}",
     "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
    }
   }
  ]
 }
}
```

Exemple 2

L'exemple suivant illustre comment ajouter la fonction sourceip () en tant que propriété utilisateur MQTT à l'aide de modèles de substitution.

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "republish": {
```

```
      "topic": "${topic()}/republish",
      "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
      "headers": {
       "payloadFormatIndicator": "UTF8_DATA",
       "contentType": "rule/contentType",
       "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
       "userProperties": [
        {
         "key": "ruleKey1",
         "value": "ruleValue1"
        },
        {
         "key": "sourceip",
         "value": "${sourceip()}"
        }
       ]
      }
     }
    }
   ]
  }
}
```

Vous pouvez récupérer l'adresse IP source à partir des messages transmis aux AWS IoT Core règles depuis les chemins Message Broker et Basic Ingest. Vous pouvez également récupérer l'adresse IP source pour IPv4 les deux IPv6 messages. L'adresse IP source sera affichée comme suit :

IPv6: `yyyy:yyyy:yyyy::yyyy:yyyy`

IPv4: `xxx.xxx.xxx.xxx`

> **Note**
>
> L'adresse IP source d'origine ne sera pas transmise par lebiais de Republier l'action..

substring(String, Int[, Int])

Prévoit un argument `String` suivi par une ou deux valeurs `Int`. Pour un argument `String` et un seul argument `Int`, cette fonction renvoie la sous-chaîne de l'argument `String` fourni provenant de l'index (de base 0, inclus) `Int` fourni à la fin de l'argument `String`. Pour un argument `String` et

deux arguments `Int`, cette fonction renvoie la sous-chaîne de l'argument `String` fourni provenant du premier argument d'index `Int` (de base 0, inclus) dans le deuxième argument d'index `Int` (de base 0, inclus). Les index qui sont inférieurs à zéro sont définis sur zéro. Les index qui sont supérieurs à la longueur de `String` sont définis sur la longueur de `String`. Pour la version des trois arguments, si le premier index est supérieur (ou égale) au deuxième index, le résultat et vide `String`.

Si les arguments fournis ne sont pas (*`String`*,*`Int`*) ou (*`String`*,*`Int`*,*`Int`*), les conversions standard sont appliquées aux arguments pour tenter de les convertir dans les types corrects. Si les types ne peuvent pas être convertis, le résultat de la fonction est `Undefined`. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`substring("012345", 0)` = "012345".

`substring("012345", 2)` = "2345".

`substring("012345", 2.745)` = "2345".

`substring(123, 2)` = "3".

`substring("012345", -1)` = "012345".

`substring(true, 1.2)` = "true".

`substring(false, -2.411E247)` = "false".

`substring("012345", 1, 3)` = "12".

`substring("012345", -50, 50)` = "012345".

`substring("012345", 3, 1)` = "".

sql_version()

Renvoie la version SQL spécifiée dans cette règle. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`sql_version()` = "2016-03-23"

sqrt(Decimal)

Renvoie la racine carrée d'un nombre en radians. Les arguments `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `sqrt(9)` = 3.0.

Type d'argument	Résultat
`Int`	La racine carrée de l'argument.
`Decimal`	La racine carrée de l'argument.
`Boolean`	`Undefined .`
`String`	La racine carrée de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined .`
Tableau	`Undefined .`
Objet	`Undefined .`
Null	`Undefined .`
Non défini	`Undefined .`

startswith(String, String)

Renvoie une valeur `Boolean` si le premier argument de type chaîne commence par le deuxième argument de type chaîne. Si l'un des arguments est `Null` ou `Undefined`, le résultat a la valeur `Undefined`. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`startswith("ranger","ran")` = true

Type d'argument 1	Type d'argument 2	Résultat
`String`	`String`	Si la première chaîne commenc deuxième chaîne.
Autre valeur	Autre valeur	Les deux arguments sont conve chaînes à l'aide des règles de c n standard. Renvoie la valeur tr la première chaîne commence p deuxième chaîne. Si l'un des ar est `Null` ou `Undefined` , le ré valeur `Undefined` .

tan(Decimal)

Renvoie la tangente d'un nombre en radians. Les valeurs `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `tan(3)` = -0.1425465430742778

Type d’argument	Résultat
`Int`	`Decimal` (avec double précision), la tangente de l'argument.
`Decimal`	`Decimal` (avec double précision), la tangente de l'argument.
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), la tangente de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Tableau	`Undefined` .

Type d’argument	Résultat
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

tanh(Decimal)

Renvoie la tangente hyperbolique d'un nombre en radians. Les valeurs `Decimal` sont arrondis pour une meilleure prévision avant l'application de la fonction. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple : `tanh(2.3)` = 0,9800963962661914

Type d’argument	Résultat
`Int`	`Decimal` (avec double précision), la tangente hyperbolique de l'argument.
`Decimal`	`Decimal` (avec double précision), la tangente hyperbolique de l'argument.
`Boolean`	`Undefined` .
`String`	`Decimal` (avec double précision), la tangente hyperbolique de l'argument. Si la chaîne ne peut pas être convertie en une valeur `Decimal`, le résultat est `Undefined` .
Tableau	`Undefined` .
Objet	`Undefined` .
Null	`Undefined` .
Non défini	`Undefined` .

time_to_epoch (Chaîne, Chaîne)

Utilisez cette fonction `time_to_epoch` pour convertir une chaîne d'horodatage en un nombre de millisecondes en temps d'époque Unix. Pris en charge par SQL 2016-03-23 et versions ultérieures. Pour convertir des millisecondes en une chaîne d'horodatage formatée, veuillez consulter parse_time(String, Long[, String]).

La fonction `time_to_epoch` attend les arguments suivants :

timestamp

(Chaîne) Chaîne d'horodatage à convertir en millisecondes depuis l'ère Unix. Si la chaîne d'horodatage ne spécifie pas de fuseau horaire, la fonction utilise le fuseau horaire UTC.

pattern

(Chaîne) Un date/time modèle qui suit les formats JDK11 temporels.

Exemples :

`time_to_epoch("2020-04-03 09:45:18 UTC+01:00", "yyyy-MM-dd HH:mm:ss VV")=` 1585903518000

`time_to_epoch("18 December 2015", "dd MMMM yyyy")=` 1450396800000

`time_to_epoch("2007-12-03 10:15:30.592 America/Los_Angeles", "yyyy-MM-dd HH:mm:ss.SSS z")=` 1196705730592

timestamp()

Renvoie l'horodatage actuel en millisecondes à partir de 00:00:00 Temps universel coordonné (UTC), jeudi 1er janvier 1970, tel qu'observé par le moteur de règles. AWS IoT Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple: `timestamp()` = 1481825251155

topic(Decimal)

Il renvoie la rubrique vers laquelle le message qui a déclenché la règle a été envoyé. Si aucun paramètre n'est indiqué, la rubrique entière est renvoyée. Le paramètre `Decimal` est utilisé pour

spécifier un segment de rubrique spécifique, avec le chiffre 1 désignant le premier segment. Pour la rubrique `foo/bar/baz`, topic(1) renvoie `foo`, topic(2) renvoie `bar`, et ainsi de suite. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`topic()` = "things/myThings/thingOne"

`topic(1)` = "things"

Lorsque Basic Ingest est utilisé, le préfixe initial de la rubrique (`$aws/rules/`*`rule-name`*) n'est pas disponible pour la fonction topic(). Prenons l'exemple de la rubrique suivante :

`$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights`

`topic()` = "Buildings/Building5/Floor2/Room201/Lights"

`topic(3)` = "Floor2"

traceid()

Renvoie l'ID de suivi (UUID) du message MQTT ou une valeur `Undefined` si le message n'a pas été pas envoyé via MQTT. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`traceid()` = "12345678-1234-1234-1234-123456789012"

transformation (chaîne, objet, tableau)

Renvoie un tableau d'objets contenant le résultat de la transformation spécifiée du paramètre `Object` sur le paramètre `Array`.

Pris en charge par SQL 2016-03-23 et versions ultérieures.

Chaîne

> Le mode de transformation à utiliser. Reportez-vous au tableau suivant pour connaître les modes de transformation pris en charge et la manière dont ils créent le `Result` à partir des paramètres `Object` et `Array`.

Objet

Un objet qui contient les attributs à appliquer à chaque élément du `Array`.

Tableau

Tableau d'objets auxquels les attributs de `Object` sont appliqués.

Chaque objet de ce tableau correspond à un objet dans la réponse de la fonction. Chaque objet de la réponse de la fonction contient les attributs présents dans l'objet d'origine et les attributs fournis par `Object` tels que déterminés par le mode de transformation spécifié dans `String`.

`String` paramètre	**`Object`** paramètre	**`Array`** paramètre	Résultat
`enrichArray`	Objet	Tableau d'objets	Tableau d'objets dans lequel chaque objet contient les attributs d'un élément du paramètre `Array` et les attributs du paramètre `Object`.
Toute autre valeur	N'importe quelle valeur	N'importe quelle valeur	Non défini

> **Note**
>
> Le tableau renvoyé par cette fonction est limité à 128 KiB.

Exemple 1 de fonction de transformation

Cet exemple montre comment la fonction transform() produit un tableau unique d'objets à partir d'un objet de données et d'un tableau.

Dans cet exemple, le message suivant est publié dans la rubrique MQTT `A/B`.

```
{
```

```
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

Cette instruction SQL pour une action de règle de rubrique utilise la fonction transform() avec une valeur `String` de `enrichArray`. Dans cet exemple, `Object` est la propriété `attributes` de la charge utile du message et `Array` est le tableau `values`, qui contient trois objets.

```
select value transform("enrichArray", attributes, values) from 'A/B'
```

À la réception de la charge utile du message, l'instruction SQL donne la réponse suivante.

```
[
  {
    "a": 3,
    "data1": 1,
    "data2": 2
  },
  {
    "b": 4,
    "data1": 1,
    "data2": 2
  },
  {
    "c": 5,
    "data1": 1,
    "data2": 2
  }
]
```

Exemple 2 de fonction de transformation

Cet exemple montre comment la fonction transform() peut utiliser des valeurs littérales pour inclure et renommer des attributs individuels à partir de la charge utile du message.

Dans cet exemple, le message suivant est publié dans la rubrique MQTT `A/B`. Il s'agit du même message que celui utilisé dans the section called "Exemple 1 de fonction de transformation".

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

Cette instruction SQL pour une action de règle de rubrique utilise la fonction transform() avec une valeur `String` de `enrichArray`. Le `Object` dans la fonction transform() possède un seul attribut nommé `key` avec la valeur de `attributes.data1` dans la charge utile du message et `Array` est le tableau `values` qui contient les trois mêmes objets que ceux utilisés dans l'exemple précédent.

```
select value transform("enrichArray", {"key": attributes.data1}, values) from 'A/B'
```

À la réception de la charge utile du message, cette instruction SQL donne la réponse suivante. Notez comment la propriété `data1` est nommée `key` dans la réponse.

```
[
  {
    "a": 3,
    "key": 1
  },
```

```
  {
    "b": 4,
    "key": 1
  },
  {
    "c": 5,
    "key": 1
  }
]
```

Exemple 3 de fonction de transformation

Cet exemple montre comment la fonction transform() peut être utilisée dans des clauses SELECT imbriquées pour sélectionner plusieurs attributs et créer de nouveaux objets pour un traitement ultérieur.

Dans cet exemple, le message suivant est publié dans la rubrique MQTT `A/B`.

```
{
  "data1": "example",
  "data2": {
    "a": "first attribute",
    "b": "second attribute",
    "c": [
      {
        "x": {
          "someInt": 5,
          "someString": "hello"
        },
        "y": true
      },
      {
        "x": {
          "someInt": 10,
          "someString": "world"
        },
        "y": false
      }
    ]
  }
}
```

Le `Object` pour cette fonction de transformation est l'objet renvoyé par l'instruction SELECT, qui contient les éléments a et b de l'objet `data2` du message. Le paramètre `Array` comprend les deux objets du tableau `data2.c` figurant dans le message d'origine.

```
select value transform('enrichArray', (select a, b from data2), (select value c from
 data2)) from 'A/B'
```

Avec le message précédent, l'instruction SQL donne la réponse suivante.

```
[
  {
    "x": {
      "someInt": 5,
      "someString": "hello"
    },
    "y": true,
    "a": "first attribute",
    "b": "second attribute"
  },
  {
    "x": {
      "someInt": 10,
      "someString": "world"
    },
    "y": false,
    "a": "first attribute",
    "b": "second attribute"
  }
]
```

Le tableau renvoyé dans cette réponse peut être utilisé avec des actions de règles de rubrique qui prennent en charge `batchMode`.

trim(String)

Supprime tous les espaces de début et de fin de la valeur `String` fournie. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemple :

`Trim(" hi ")` = "bonjour"

Type d'argument	Résultat
`Int`	La représentation `String` de `Int` avec tous les espaces de début et de fin supprimés.
`Decimal`	La représentation `String` de `Decimal` avec tous les espaces de début et de fin supprimés.
`Boolean`	La représentation `String` de la valeur `Boolean` (« vrai » ou « faux ») avec tous les espaces de début et de fin supprimés.
`String`	L'argument `String` avec tous les espaces de début et de fin supprimés.
Tableau	La représentation `String` de la valeur `Array` à l'aide des règles de conversion standard.
Objet	La représentation `String` de l'objet à l'aide des règles de conversion standard.
Null	`Undefined` .
Non défini	`Undefined` .

trunc(Decimal, Int)

Tronque le premier argument du nombre de `Decimal`, spécifié par le deuxième argument. Si le deuxième argument est inférieur à zéro, il est défini sur zéro. Si le deuxième argument est supérieur à 34, il est défini sur 34. Les zéros de fin sont supprimés du résultat. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`trunc(2.3, 0)` = 2.

`trunc(2.3123, 2)` = 2.31.

`trunc(2.888, 2)` = 2.88.

`trunc(2.00, 5)` = 2.

Type d'argument 1	Type d'argument 2	Résultat
`Int`	`Int`	La valeur source.
`Int/Decimal`	`Int/Decimal`	Le premier argument est tronqu longueur décrite par le deuxièm Le deuxième argument, s'il ne s d'un `Int`, est arrondi à la valeur inférieure la plus proche.
`Int/Decimal/String`	`Int/Decimal`	Le premier argument est tronqu la longueur décrite par le deuxiè argument. Le deuxième argume ne s'agit pas d'un `Int`, est arron valeur `Int` inférieure la plus pro valeur `String` est convertie en `Decimal`. Si la chaîne ne peut convertie, le résultat est `Undefi`
Autre valeur		`Undefined` .

upper(String)

Renvoie la version en majuscules de la valeur `String` donnée. Les arguments non-`String` sont convertis en valeurs `String` à l'aide des règles de conversion standard. Prise en charge par SQL 2015-10-08 et versions ultérieures.

Exemples :

`upper("hello")` = "BONJOUR"

`upper(["hello"])` = "[\"BONJOUR\"]"

Littéraux

Vous pouvez spécifier directement des objets littéraux dans les clauses SELECT et WHERE de votre règle SQL, qui permet de transmettre des informations.

> **Note**
>
> Les littéraux sont disponibles uniquement lors de l'utilisation de SQL 2016-03-23 ou versions ultérieures.

Une syntaxe d'objet JSON est utilisée (paires clé-valeur, séparées par des virgules, où les clés sont des chaînes, et les valeurs des valeurs JSON, entourées d'accolades {}). Par exemple :

Charge utile entrante publiée dans une rubrique `topic/subtopic` : `{"lat_long": [47.606,-122.332]}`

Instruction SQL : `SELECT {'latitude': get(lat_long, 0),'longitude':get(lat_long, 1)} as lat_long FROM 'topic/subtopic'`

La charge utile sortante résultante serait : `{"lat_long": {"latitude":47.606,"longitude":-122.332}}`.

Vous pouvez également spécifier des tableaux dans des clauses SELECT et WHERE de votre règle SQL, qui vous permet de regrouper des informations. Une syntaxe JSON est utilisée (éléments séparés par des virgules entre crochets [] pour créer un littéral de tableau). Par exemple :

Charge utile entrante publiée dans une rubrique `topic/subtopic` : `{"lat": 47.696, "long": -122.332}`

Instruction SQL : `SELECT [lat,long] as lat_long FROM 'topic/subtopic'`

La charge utile de sortie serait : `{"lat_long": [47.606,-122.332]}`.

Instructions Case

Les instructions case peuvent être utilisées pour l’exécution de branche, comme une instruction switch.

Syntaxe :

```
CASE v WHEN t[1] THEN r[1]
  WHEN t[2] THEN r[2] ...
  WHEN t[n] THEN r[n]
  ELSE r[e] END
```

L'expression *v* est évaluée et fait l'objet d'une comparaison d'égalité avec la valeur *t[i]* de chaque clause WHEN. Si une correspondance est trouvée, l'expression *r[i]* correspondante devient le résultat de l'instruction CASE. Les clauses WHEN sont évaluées de telle sorte que s'il existe plusieurs clauses correspondantes, le résultat de la première clause correspondante devient le résultat de l'instruction CASE. S'il n'y a aucune correspondance, le résultat *r[e]* de la clause ELSE est le résultat. S'il n'y a ni correspondance ni clause ELSE, le résultat est Undefined.

Les instructions CASE requièrent au moins une clause WHEN. Une clause ELSE est facultative.

Par exemple :

Charge utile entrante publiée dans une rubrique `topic/subtopic` :

```
{
    "color":"yellow"
}
```

Instruction SQL :

```
SELECT CASE color
        WHEN 'green' THEN 'go'
        WHEN 'yellow' THEN 'caution'
        WHEN 'red' THEN 'stop'
        ELSE 'you are not at a stop light' END as instructions
    FROM 'topic/subtopic'
```

La charge utile de sortie serait :

```
{
    "instructions":"caution"
}
```

> **Note**
>
> Si *v* est Undefined, le résultat de l'instruction case est Undefined.

Extensions JSON

Vous pouvez utiliser les extensions suivantes de la syntaxe SQL ANSI pour faciliter l'utilisation d'objets JSON imbriqués.

« . » Opérateur

Cet opérateur accède aux membres dans les objets JSON intégrés et fonctionne de la même manière que le SQL ANSI et. JavaScript Par exemple :

```
SELECT foo.bar AS bar.baz FROM 'topic/subtopic'
```

sélectionne la valeur de la propriété `bar` dans l'objet `foo` à partir de la charge utile du message suivant envoyé à la rubrique `topic/subtopic`.

```
{
  "foo": {
    "bar": "RED",
    "bar1": "GREEN",
    "bar2": "BLUE"
  }
}
```

Si le nom d'une propriété JSON inclut un trait d'union ou des caractères numériques, la notation « point » ne fonctionnera pas. Vous devez plutôt utiliser la fonction get pour extraire la valeur de la propriété.

Dans cet exemple, le message suivant est envoyé dans la rubrique `iot/rules`.

```
{
  "mydata": {
    "item2": {
      "0": {
        "my-key": "myValue"
      }
    }
  }
}
```

Normalement, la valeur de `my-key` serait identifiée comme dans cette requête.

```
SELECT * from iot/rules WHERE mydata.item2.0.my-key= "myValue"
```

Toutefois, étant donné que le nom de la propriété `my-key` contient un trait d'union et `item2` un caractère numérique, la fonction get doit être utilisée comme le montre la requête suivante.

```
SELECT * from 'iot/rules' WHERE get(get(get(mydata,"item2"),"0"),"my-key") = "myValue"
```

* Opérateur

Cela fonctionne de la même manière que le caractère générique * dans SQL ANSI. Il est utilisé dans la clause SELECT uniquement et crée un nouvel objet JSON contenant les données du message. Si la charge utile du message n'est pas au format JSON, * renvoie la charge utile du message entier sous la forme d'octets bruts. Par exemple :

```
SELECT * FROM 'topic/subtopic'
```

Appliquer une fonction à une valeur d'attribut

Voici un exemple de charge utile JSON qui peut être publiée par un appareil :

```
{
    "deviceid" : "iot123",
    "temp" : 54.98,
    "humidity" : 32.43,
    "coords" : {
        "latitude" : 47.615694,
        "longitude" : -122.3359976
    }
}
```

L'exemple suivant applique une fonction à une valeur d'attribut dans une charge utile JSON :

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

Le résultat de cette requête est l'objet JSON suivant :

```
{
   "temp": 54.98,
   "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"
}
```

Modèles de substitution

Vous pouvez utiliser un modèle de substitution pour augmenter les données JSON renvoyées lorsqu'une règle est déclenchée et AWS IoT exécute une action. La syntaxe d'un modèle de substitution est ${ expression}, où l'expression peut être n'importe quelle expression prise AWS IoT en charge par les clauses SELECT, WHERE etAWS IoT actions liées aux règles. Cette expression peut être connectée à un champ d'action d'une règle, ce qui vous permet de configurer dynamiquement une action. En effet, cette fonctionnalité remplace un élément d'information dans une action. Cela inclut les fonctions, les opérateurs et les informations présents dans la charge utile du message d'origine.

> **Important**
>
> Puisqu'une expression dans un modèle de substitution est évaluée séparément de l'instruction « SELECT... », vous ne pouvez pas référencer un alias créé à l'aide de la clause AS. Vous pouvez référencer uniquement les informations présentes dans la charge utile d'origine, fonctions et opérateurs.

Pour plus d'informations concernant les expressions prises en charge, consultez la page AWS IoT Référence SQL.

Les actions de règle suivantes prennent en charge les modèles de substitution. Chaque action prend en charge différents champs qui peuvent être substitués.

- Apache Kafka
- CloudWatch alarmes
- CloudWatch Journaux
- CloudWatch métriques
- DynamoDB
- Dynamo DBv2
- Elasticsearch
- HTTP
- IoT Analytics
- AWS IoT Events

- AWS IoT SiteWise
- Kinesis Data Streams
- Firehose
- Lambda
- Emplacement
- OpenSearch
- Republish
- S3
- SNS
- SQS
- Step Functions
- Timestream

Les modèles de substitution apparaissent dans les paramètres d'action au sein d'une règle :

```
{
    "sql": "SELECT *, timestamp() AS timestamp FROM 'my/iot/topic'",
    "ruleDisabled": false,
    "actions": [{
        "republish": {
            "topic": "${topic()}/republish",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

Si cette règle est déclenchée par le code JSON suivant publié dans `my/iot/topic` :

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    }
}
```

Cette règle publie ensuite le code JSON suivant sur`my/iot/topic/republish`, qui AWS IoT remplace : `${topic()}/republish`

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    },
    "timestamp": 1579637878451
}
```

Requêtes d'objets imbriqués

Vous pouvez utiliser des clauses SELECT imbriquées pour interroger les attributs dans les tableaux et les objets JSON internes. Pris en charge par SQL 2016-03-23 et versions ultérieures.

Examinez le message MQTT suivant :

```
{
    "e": [
        { "n": "temperature", "u": "Cel", "t": 1234, "v": 22.5 },
        { "n": "light", "u": "lm", "t": 1235, "v": 135 },
        { "n": "acidity", "u": "pH", "t": 1235, "v": 7 }
    ]
}
```

Example

Vous pouvez convertir des valeurs en un nouveau tableau avec la règle suivante.

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

La règle génère le résultat suivant.

```
{
    "sensors": [
        "temperature",
        "light",
        "acidity"
```

```
    ]
}
```

Example

En utilisant le même message MQTT, vous pouvez également interroger une valeur spécifique dans un objet imbriqué avec la règle suivante.

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

La règle génère le résultat suivant.

```
{
    "temperature": [
        {
            "v": 22.5
        }
    ]
}
```

Example

Vous pouvez également aplatir la sortie avec une règle plus compliquée.

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

La règle génère le résultat suivant.

```
{
    "temperature": 22.5
}
```

Utilisation des charges utiles binaires

Pour traiter la charge utile de votre message comme des données binaires brutes (plutôt que comme un objet JSON), vous pouvez utiliser l'opérateur * pour y faire référence dans une clause SELECT.

Dans cette rubrique :

- Exemples de charge utile binaire
- Décodage des charges utiles des messages protobuf

Exemples de charge utile binaire

Lorsque vous utilisez * pour désigner la charge utile du message sous forme de données binaires brutes, vous pouvez ajouter des données à la règle. Si vous avez une charge utile vide ou une charge utile JSON, des données peuvent être ajoutées à la charge utile résultante à l'aide de la règle. Vous trouverez ci-dessous des exemples de clauses `SELECT` prises en charge.

- Vous pouvez utiliser les clauses `SELECT` suivantes avec uniquement un * pour les charges utiles binaires.
  - ```
    SELECT * FROM 'topic/subtopic'
    ```
  - ```
    SELECT * FROM 'topic/subtopic' WHERE timestamp() % 12 = 0
    ```
- Vous pouvez également ajouter des données et utiliser les clauses `SELECT` suivantes.
  - ```
    SELECT *, principal() as principal, timestamp() as time FROM 'topic/subtopic'
    ```
  - ```
    SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'topic/subtopic'
    ```
- Vous pouvez également utiliser ces clauses `SELECT` avec des charges utiles binaires.
  - Ce qui suit fait référence à `device_type` dans la clause WHERE.

    ```
    SELECT * FROM 'topic/subtopic' WHERE device_type = 'thermostat'
    ```
  - Les éléments suivants sont pris en charge.

    ```
    {
     "sql": "SELECT * FROM 'topic/subtopic'",
     "actions": [
      {
       "republish": {
        "topic": "device/${device_id}"
       }
      }
     ]
    }
    ```

Les actions de règles suivantes ne prennent pas en charge les charges utiles binaires. Vous devez donc les décoder.

- Certaines actions de règles ne prennent pas en charge les données utiles binaires, comme les actions Lambda, et vous devez donc décoder les données utiles binaires. L'action de la règle Lambda peut recevoir des données binaires si elles sont codées en base64 et placées dans une charge utile JSON. Pour cela, apportez les modifications suivantes à la règle :

```
SELECT encode(*, 'base64') AS data FROM 'my_topic'
```

- L'instruction SQL ne prend pas en charge les chaînes en entrée. Pour convertir une chaîne en JSON, vous pouvez exécuter la commande suivante.

```
SELECT decode(encode(*, 'base64'), 'base64') AS payload FROM 'topic'
```

Décodage des charges utiles des messages protobuf

Protocol Buffers (protobuf) est un format de données open source utilisé pour sérialiser des données structurées sous une forme binaire compacte. Il est utilisé pour transmettre des données sur des réseaux ou pour les stocker dans des fichiers. Protobuf vous permet d'envoyer des données sous forme de petits paquets et à un rythme plus rapide que les autres formats de messagerie. AWS IoT Core Les règles prennent en charge protobuf en fournissant la fonction SQL decode (value, decodingScheme), qui vous permet de décoder les charges utiles des messages codés par protobuf au format JSON et de les acheminer vers les services en aval. Cette section détaille le step-by-step processus de configuration du décodage protobuf dans Rules. AWS IoT Core

Dans cette section :

- Prérequis
- Création de fichiers descripteurs
- Charger les fichiers descripteurs dans un compartiment S3
- Configurer le décodage protobuf dans Règles
- Limites
- Bonnes pratiques

Prérequis

- Compréhension de base de Protocol Buffers (protobuf)
- Les `.proto`fichiers qui définissent les types de messages et les dépendances associées

- Installation de Protobuf Compiler (protoc) sur votre système

Création de fichiers descripteurs

Si vous disposez déjà de fichiers descripteurs, vous pouvez ignorer cette étape. Un fichier descripteur (`.desc`) est une version compilée d'un fichier `.proto`, qui est un fichier texte qui définit les structures de données et les types de messages à utiliser dans une sérialisation protobuf. Pour générer un fichier descripteur, vous devez définir un fichier `.proto` et utiliser le compilateur protoc pour le compiler.

1. Créez des fichiers `.proto` qui définissent les types de messages. Un fichier `.proto` manifeste peut ressembler à l'exemple suivant :

   ```
   syntax = "proto3";

   message Person {
     optional string name = 1;
     optional int32 id = 2;
     optional string email = 3;
   }
   ```

   Dans cet exemple de fichier `.proto`, vous utilisez la syntaxe proto3 et définissez le type de message `Person`. La définition du message `Person` spécifie trois champs (nom, identifiant et e-mail). Pour plus d'informations sur les formats de message de fichier `.proto`, veuillez consulter le Guide des langues (proto3).

2. Utilisez le compilateur protoc pour compiler les fichiers `.proto` et générer un fichier de descripteurs. La création d'un fichier (`.desc`) descripteur est illustrée ci-dessous :

   ```
   protoc --descriptor_set_out=<FILENAME>.desc \
       --proto_path=<PATH_TO_IMPORTS_DIRECTORY> \
       --include_imports \
       <PROTO_FILENAME>.proto
   ```

   Cet exemple de commande génère un fichier descripteur que `<FILENAME>.desc` AWS IoT Core Rules peut utiliser pour décoder les charges utiles protobuf conformes à la structure de données définie dans. `<PROTO_FILENAME>.proto`

   - `--descriptor_set_out`

Spécifie le nom du fichier (<FILENAME>.desc) descripteur qui doit être généré.

- --proto_path

  Spécifie l'emplacement de tous les fichiers .proto importés référencés par le fichier en cours de compilation. Vous pouvez spécifier l'indicateur plusieurs fois si vous avez plusieurs fichiers .proto importés avec des emplacements différents.

- --include_imports

  Spécifie que tous les fichiers .proto importés doivent également être compilés et inclus dans le fichier <FILENAME>.desc descripteur.

- <PROTO_FILENAME>.proto

  Spécifie le nom du fichier .proto que vous souhaitez compiler.

Pour de plus amples informations sur la référence protoc, veuillez consulter Référence de l'API.

Charger les fichiers descripteurs dans un compartiment S3

Après avoir créé vos fichiers descripteurs<FILENAME>.desc, chargez-les dans un compartiment Amazon S3 <FILENAME>.desc à l'aide de l' AWS API, du AWS SDK ou du. AWS Management Console

Considérations Importantes

- Assurez-vous de télécharger les fichiers descripteurs dans un compartiment Amazon S3 situé dans le même compartiment que celui Compte AWS dans Région AWS lequel vous souhaitez configurer vos règles.
- Assurez-vous d'autoriser l' AWS IoT Core accès pour lire le contenu FileDescriptorSet de S3. Si le chiffrement côté serveur (SSE) est désactivé dans votre compartiment S3 ou s'il est chiffré à l'aide de clés gérées par Amazon S3 (SSE-S3), aucune configuration de stratégie supplémentaire n'est requise. Cela peut être accompli à l'aide de l'exemple de politique de compartiment :

  JSON

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

```
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
     "Service": "iot.amazonaws.com"
    },
    "Action": "s3:Get*",
                      "Resource": "arn:aws:s3:::<BUCKET NAME>/<FILENAME>.desc"
  }
 ]
}
```

- Si votre compartiment S3 est chiffré à l'aide d'une AWS Key Management Service clé (SSE-KMS), assurez-vous d' AWS IoT Core autoriser l'utilisation de la clé lors de l'accès à votre compartiment S3. Pour ce faire, vous pouvez ajouter cette déclaration à votre stratégie de clé :

```
{
 "Sid": "Statement1",
 "Effect": "Allow",
 "Principal": {
  "Service": "iot.amazonaws.com"
 },
 "Action": [
  "kms:Decrypt",
  "kms:GenerateDataKey*",
  "kms:DescribeKey"
 ],
        "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

}
```

Configurer le décodage protobuf dans Règles

Après avoir chargé les fichiers descripteurs dans votre compartiment Amazon S3, configurez une Règle capable de décoder le format de charge utile de votre message protobuf à l'aide de la fonction SQL decode (value, decodingScheme). Une signature de fonction détaillée et un exemple se trouvent dans la fonction SQL decode(value, decodingScheme) de la AWS IoT référence SQL.

Voici un exemple d'expression SQL utilisant la fonction decode (value, decodingScheme) :

```
SELECT VALUE decode(*, 'proto', '<BUCKET NAME>', '<FILENAME>.desc', '<PROTO_FILENAME>',
 '<PROTO_MESSAGE_TYPE>') FROM '<MY_TOPIC>'
```

Dans cet exemple d'expression :

- Vous utilisez la fonction SQL decode (value, decodingScheme) pour décoder la charge utile du message binaire référencée par `*`. Il peut s'agir d'une charge utile binaire codée en protobuf ou d'une chaîne JSON représentant une charge utile protobuf codée en base64.
- La charge utile du message fournie est codée à l'aide du type de message `Person` défini dans `PROTO_FILENAME.proto`.
- Le compartiment Amazon S3 nommé `BUCKET NAME` contient le `FILENAME.desc` généré à partir de `PROTO_FILENAME.proto`.

Une fois la configuration terminée, publiez un message AWS IoT Core sur le sujet auquel la règle est abonnée.

Limites

AWS IoT Core Les règles prennent en charge protobuf avec les limitations suivantes :

- Le décodage des charges utiles des messages protobuf dans les modèles de substitution n'est pas pris en charge.
- Lorsque vous décodez les charges utiles des messages protobuf, vous pouvez utiliser la fonction SQL de décodage dans une seule expression SQL jusqu'à deux fois.
- La taille maximale de la charge utile entrante est de 128 KiB (1 KiB = 1024 octets), la taille maximale de la charge utile sortante est de 128 KiB et la taille maximale d'un objet `FileDescriptorSet` stocké dans un compartiment Amazon S3 est de 32 KiB.
- Les compartiments Amazon S3 chiffrés avec le chiffrement SSE-C ne sont pas pris en charge.

Bonnes pratiques

Voici quelques bonnes pratiques et des conseils de dépannage.

- Sauvegardez vos fichiers proto dans le compartiment Amazon S3.

  Il est recommandé de sauvegarder vos fichiers proto en cas de problème. Par exemple, si vous modifiez incorrectement les fichiers proto sans sauvegarde lors de l'exécution de protoc, cela

peut entraîner des problèmes dans votre stack de production. Il existe plusieurs méthodes pour sauvegarder vos fichiers dans un compartiment Amazon S3. Par exemple, vous pouvez utiliser la gestion des versions dans les compartiments S3. Pour plus d'informations sur la sauvegarde de fichiers dans des compartiments Amazon S3, veuillez consulter le manuel du développeur Amazon S3.

- Configurez la AWS IoT journalisation pour afficher les entrées du journal.

  Il est recommandé de configurer la AWS IoT journalisation de manière à pouvoir consulter AWS IoT les journaux de votre compte CloudWatch. Lorsque la requête SQL d'une règle appelle une fonction externe, AWS IoT Core Rules génère une entrée de journal avec un `eventType` de`FunctionExecution`, qui contient le champ de raison qui vous aidera à résoudre les problèmes d'échec. Les erreurs possibles incluent un objet Amazon S3 introuvable ou un descripteur de fichier protobuf non valide. Pour plus d'informations sur la façon de configurer la journalisation AWS IoT et de consulter les entrées du journal, veuillez consulter Configurer la AWS IoT journalisation et les entrées du journal du moteur de règles.
- Effectuez la mise à jour de `FileDescriptorSet` à l'aide d'une nouvelle clé d'objet et mettez à jour la clé d'objet dans votre règle.

  Vous pouvez effectuer une mise à jour de `FileDescriptorSet` en chargeant un fichier descripteur mis à jour dans votre compartiment Amazon S3. Vos mises à jour de `FileDescriptorSet` peuvent prendre jusqu'à 15 minutes pour être prises en compte. Pour éviter ce retard, il est recommandé de télécharger votre `FileDescriptorSet` mis à jour à l'aide d'une nouvelle clé d'objet et de mettre à jour la clé d'objet dans votre Règle.

Versions de SQL

Le moteur de AWS IoT règles utilise une syntaxe de type SQL pour sélectionner les données des messages MQTT. Les instructions SQL sont interprétées en fonction d'une version SQL spécifiée avec la propriété `awsIotSqlVersion` dans un document JSON qui décrit la règle. Pour plus d'informations sur la structure des documents de règle JSON, consultez Création d'une règle. La `awsIotSqlVersion` propriété vous permet de spécifier la version du moteur de règles AWS IoT SQL que vous souhaitez utiliser. Lorsqu'une nouvelle version est déployée, vous pouvez continuer à utiliser une version antérieure ou modifier votre règle pour utiliser la nouvelle version. Vos règles actuelles continuent d'utiliser la version avec laquelle elles ont été créées.

L'exemple de code JSON suivant montre comment spécifier la version SQL à l'aide la propriété `awsIotSqlVersion`.

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "republish": {
            "topic": "my-mqtt-topic",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

AWS IoT prend actuellement en charge les versions SQL suivantes :

- `2016-03-23` – La version SQL construite le 23/03/2016 (recommandée).
- `2015-10-08` – La version de SQL d'origine créée le 08-10-2015.
- `beta` – La version bêta de SQM la plus récente. Cette version pourrait introduire des modifications radicales dans vos règles.

Nouveautés de la version 2016-03-23 du moteur de règles SQL

- Correctifs pour sélectionner les objets JSON imbriqués.
- Correctifs pour les requêtes de tableaux.
- Prise en charge des requêtes inter-objet. Pour de plus amples informations, veuillez consulter Requêtes d'objets imbriqués.
- Prise en charge de la génération en sortie d'un tableau comme un objet de niveau supérieur.
- Ajout de la fonction `encode(value, encodingScheme)`, qui peut être appliquée sur les données de format JSON et non-JSON. Pour plus d'informations, consultez la fonction d'encodage.

Générer un **`Array`** en sortie comme un objet de niveau supérieur

Cette fonction permet à une règle de retourner un tableau comme un objet de niveau supérieur. Par exemple, avec le message MQTT suivant :

```
{
    "a": {"b":"c"},
    "arr":[1,2,3,4]
```

```
}
```

Et la règle suivante :

```
SELECT VALUE arr FROM 'topic'
```

La règle génère le résultat suivant.

```
[1,2,3,4]
```

AWS IoT Service Device Shadow

Le service AWS IoT Device Shadow ajoute des ombres AWS IoT aux objets. Les ombres peuvent rendre l'état d'un appareil accessible aux applications et à d'autres services, que l'appareil soit connecté AWS IoT ou non. AWS IoT les objets objets peuvent avoir plusieurs ombres nommées, de sorte que votre solution IoT dispose de davantage d'options pour connecter vos appareils à d'autres applications et services.

AWS IoT les objets objets n'ont aucune ombre tant qu'ils ne sont pas créés explicitement. Les ombres peuvent être créées, mises à jour et supprimées à l'aide de la AWS IoT console. Les appareils, les autres clients Web et les services peuvent créer, mettre à jour et supprimer des ombres à l'aide de MQTT et des rubriques MQTT réservées, de HTTP à l'aide de l'API REST Device Shadow et le AWS CLI pour AWS IoT. Les ombres étant stockées AWS dans le cloud, elles peuvent collecter et rapporter des données sur l'état de l'appareil à partir d'applications et d'autres services cloud, que l'appareil soit connecté ou non.

Utilisation des shadows

Les shadows fournissent un magasin de données fiable pour que les appareils, les applications et d'autres services cloud partagent des données. Ils permettent aux appareils, aux applications et aux autres services cloud de se connecter et déconnecter sans perdre l'état d'un appareil.

Lorsque les appareils, applications et autres services cloud sont connectés AWS IoT, ils peuvent accéder à l'état actuel d'un appareil et le contrôler à travers ses ombres. Par exemple, une application peut demander la modification de l'état d'un appareil en mettant à jour une ombre. AWS IoT publie un message indiquant la modification apportée à l'appareil. L'appareil reçoit ce message, met à jour son état pour qu'il corresponde et publie un message avec son état mis à jour. Le service Device Shadow reflète cet état mis à jour dans le shadow correspondant. L'application peut s'abonner à la mise à jour du shadow ou interroger le shadow pour obtenir son état actuel.

Lorsqu'un appareil se déconnecte, une application peut toujours communiquer avec AWS IoT les ombres de l'appareil. Lorsque l'appareil se reconnecte, il reçoit l'état actuel de ses shadows pour pouvoir mettre à jour son état afin que ce dernier corresponde à celui de ses shadows, et pour pouvoir publier un message avec son état mis à jour. De même, lorsqu'une application passe hors connexion et que l'état de l'appareil change alors qu'elle est hors connexion, l'appareil conserve le shadow mis à jour afin que l'application puisse interroger les shadows pour connaître son état actuel lorsqu'elle se reconnecte.

Si vos appareils sont fréquemment hors ligne et que vous souhaitez les configurer pour recevoir des messages delta après leur reconnexion, vous pouvez utiliser la fonctionnalité de session permanente. Pour plus d'informations sur la période d'expiration des sessions persistantes, consultez la section Période d'expiration des sessions persistantes.

Choix d'utilisation de shadows nommés ou non nommés

Le service Device Shadow prend en charge les ombres nommées et non nommées, ou classiques. Un objet peut avoir plusieurs ombres nommées et pas plus d'une ombre non nommée. L'objet objet peut également avoir une ombre nommée réservée, qui fonctionne de la même manière qu'une ombre nommée, sauf que vous ne pouvez pas mettre à jour son nom. Pour plus d'informations, veuillez consulter Gestion de la mémoire réservée.

Un objet d'objet peut avoir des shadows nommés et non nommés en même temps. Toutefois, l'API utilisée pour accéder à chacun d'eux est légèrement différente. Il pourrait donc être plus efficace de décider quel type de shadow fonctionnerait le mieux pour votre solution et d'utiliser ce type uniquement. Pour de plus amples informations sur l'API permettant d'accéder aux shadows, veuillez consulter Rubriques de shadow.

Avec les shadows nommés, vous pouvez créer différentes vues de l'état d'un objet d'objet. Par exemple, vous pouvez diviser un objet d'objet avec de nombreuses propriétés en shadows avec des groupes logiques de propriétés, chacun identifié par son nom de shadow. Vous pouvez également limiter l'accès aux propriétés en les regroupant dans différents shadows et en utilisant des stratégies pour contrôler l'accès. Pour plus d'informations sur les politiques à utiliser avec les ombres des appareils, consultez la section Actions, ressources, clés de condition AWS IoT et AWS IoT Core politiques.

Les shadows classiques non nommés sont plus simples, mais un peu plus limités que les shadows nommés. Chaque AWS IoT objet ne peut avoir qu'une seule ombre sans nom. Si vous prévoyez que votre solution IoT aura un besoin limité de données de shadow, c'est peut-être ainsi que vous souhaitez commencer à utiliser les shadows. Toutefois, si vous pensez ajouter des shadows supplémentaires à l'avenir, envisagez d'utiliser des shadows nommés dès le début.

L'indexation des flottes prend en charge différemment les ombres anonymes et les ombres nommées. Pour de plus amples informations, veuillez consulter Gérer l'indexation du parc.

Accès aux shadows

Chaque shadow a une rubrique MQTT réservée et une URL HTTP qui prend en charge les actions `get`, `update` et `delete` sur le shadow.

Les shadows utilisent des documents de shadow JSON pour stocker et récupérer des données. Un document shadow contient une propriété d'état qui décrit les aspects suivants de l'état de l'appareil :

- `desired`

  Les applications spécifient les états souhaités des propriétés de l'appareil en mettant à jour l'objet `desired`.
- `reported`

  Les appareils rapportent leur état actuel dans l'objet `reported`.
- `delta`

  AWS IoT signale les différences entre l'état souhaité et l'état indiqué dans l'`delta`objet.

Les données stockées dans un shadow sont déterminées par la propriété d'état du corps du message de l'action de mise à jour. Les actions de mise à jour suivantes peuvent modifier les valeurs d'un objet de données existant, ainsi qu'ajouter et supprimer des clés et d'autres éléments dans l'objet d'état du shadow. Pour de plus amples informations sur l'accès aux shadows, veuillez consulter Utilisation des shadows sur les appareils et Utilisation des shadows dans les applications et les services.

> **Important**
>
> L'autorisation d'effectuer des demandes de mise à jour doit être limitée aux applications et aux appareils approuvés. Cela empêche la propriété d'état du shadow d'être modifiée de manière inattendue. Sinon, les appareils et les applications qui utilisent le shadow doivent être conçus de manière à s'attendre à ce que les clés figurant dans la propriété d'état changent.

Utilisation des shadows sur les appareils, dans les applications et dans d'autres services cloud

L'utilisation de shadows sur les appareils, dans les applications et dans d'autres services cloud nécessite une cohérence et une coordination entre tous ces éléments. Le service AWS IoT Device Shadow enregistre l'état d'ombre, envoie des messages lorsque l'état d'ombre change et répond aux messages qui modifient son état. Les appareils, applications et autres services cloud de votre solution IoT doivent gérer leur état et le maintenir cohérent avec l'état du shadow de l'appareil.

Les données d'état du shadow sont dynamiques et peuvent être modifiées par les appareils, les applications et les autres services cloud dotés de l'autorisation d'accéder au shadow. Pour cette raison, il est important de considérer comment chaque appareil, application et autre service cloud interagira avec le shadow. Par exemple :

- Lesappareils doivent écrire uniquement dans la propriété `reported` de l'état du shadow lors de la communication des données d'état au shadow.
- Les applications et autres services cloud doivent écrire uniquement dans la propriété `desired` lors de la communication des demandes de changement d'état à l'appareil via le shadow.

> **Important**
>
> Les données contenues dans un objet de données de shadow sont indépendantes de celles des autres shadows et des autres propriétés de l'objet d'objet, telles que les attributs d'un objet et le contenu des messages MQTT que l'appareil d'un objet d'objet peut publier. Toutefois, un appareil peut rapporter les mêmes données dans différents shadows et différentes rubriques MQTT, si nécessaire.
> Un appareil qui prend en charge plusieurs shadows doit maintenir la cohérence des données qu'il rapporte dans les différents shadows.

Ordre des messages

Rien ne garantit que les messages du AWS IoT service arriveront à l'appareil dans un ordre spécifique. Le scénario suivant montre ce qui se passe dans ce cas.

Document d'état initial :

```
{
  "state": {
    "reported": {
      "color": "blue"
    }
  },
  "version": 9,
  "timestamp": 123456776
}
```

Mise à jour 1 :

```
{
  "state": {
    "desired": {
      "color": "RED"
    }
  },
  "version": 10,
  "timestamp": 123456777
}
```

Mise à jour 2 :

```
{
  "state": {
    "desired": {
      "color": "GREEN"
    }
  },
  "version": 11,
  "timestamp": 123456778
}
```

Document d'état final :

```
{
  "state": {
    "reported": {
      "color": "GREEN"
    }
  },
  "version": 12,
  "timestamp": 123456779
}
```

Il en résulte deux messages delta :

```
{
  "state": {
    "color": "RED"
  },
  "version": 11,
```

```
  "timestamp": 123456778
}
```

```
{
  "state": {
    "color": "GREEN"
  },
  "version": 12,
  "timestamp": 123456779
}
```

L'appareil peut recevoir ces messages dans le désordre. Etant donné que l'état de ces messages est cumulé, un dispositif peut ignorer en toute sécurité tous les messages qui contiennent un numéro de version plus ancien que celui qu'il suit. Si le dispositif reçoit le delta pour la version 12 avant la version 11, il peut en toute sécurité ignorer le message concernant la version 11.

Suppression des messages de shadow

Pour réduire la taille des messages de shadow envoyés à votre appareil, définissez une règle qui sélectionne uniquement les champs dont votre appareil a besoin, puis republie le message dans une rubrique MQTT que votre appareil écoute.

La règle est spécifiée dans JSON et doit ressembler à ceci :

```
{
  "sql": "SELECT state, version FROM '$aws/things/+/shadow/update/delta'",
  "ruleDisabled": false,
  "actions": [
    {
      "republish": {
        "topic": "${topic(3)}/delta",
        "roleArn": "arn:aws:iam:123456789012:role/my-iot-role"
      }
    }
  ]
}
```

L'instruction SELECT détermine quels champs du message seront republiés dans la rubrique spécifiée. Un caractère générique « + » est utilisé pour apparier tous les noms de shadow. La règle spécifie que tous les messages correspondants doivent être republiés dans la rubrique spécifiée.

Dans ce cas, la fonction `"topic()"` est utilisée pour indiquer la rubrique dans laquelle republier. `topic(3)` correspond à la valeur du nom de l'objet dans la rubrique d'origine. Pour de plus amples informations sur la création de règles, veuillez consulter Règles pour AWS IoT.

Utilisation des shadows sur les appareils

Cette section décrit les communications entre les appareils et les ombres à l'aide de messages MQTT, la méthode préférée des appareils pour communiquer avec le service AWS IoT Device Shadow.

Les communications parallèles émulent un request/response modèle à l'aide du modèle de communication de publication/d'abonnement de MQTT. Chaque action de shadow se compose d'une rubrique de demande, d'une rubrique de réponse réussie (`accepted`) et d'une rubrique de réponse d'erreur (`rejected`).

Si vous souhaitez que les applications et les services soient en mesure de déterminer si un appareil est connecté, veuillez consulter Détection d'un appareil connecté.

> **Important**
>
> Comme MQTT utilise un modèle publish/subscribe de communication, vous devez vous abonner aux sujets de réponse avant de publier un sujet de demande. Si vous ne le faites pas, vous risquez de ne pas recevoir la réponse à la demande que vous publiez.
> Si vous utilisez un Kit SDK des appareils AWS IoTpour appeler le service Device Shadow APIs, cela est géré pour vous.

Les exemples de cette section utilisent une forme abrégée de la rubrique dans laquelle ils *`ShadowTopicPrefix`* peuvent faire référence à une ombre nommée ou non, comme décrit dans ce tableau.

Les shadows peuvent être nommés ou non (classique). Les rubriques utilisées par chacun d'eux ne diffèrent que par le préfixe de rubrique. Ce tableau indique le préfixe de rubrique utilisé par chaque type de shadow.

Valeur ***`ShadowTopicPrefix`***	Type de shadow
`$aws/things/` *`thingName`* `/shadow`	Shadow non nommé (classique)

Valeur *ShadowTopicPrefix*	Type de shadow
`$aws/things/` *thingName* `/shadow/name/` *shadowName*	Shadow nommé

> **Important**
>
> Assurez-vous que l'utilisation des shadows par votre application ou service est cohérente et prise en charge par les implémentations correspondantes sur vos appareils. Par exemple, prenez en compte la façon dont les shadows sont créés, mis à jour et supprimés. Prenez également en compte la manière dont les mises à jour sont gérées sur l'appareil et dans les applications ou services qui accèdent à l'appareil via un shadow. Votre conception doit indiquer clairement comment l'état de l'appareil est mis à jour et rapporté, et comment vos applications et services interagissent avec l'appareil et ses shadows.

Pour créer une rubrique complète, sélectionnez *ShadowTopicPrefix* pour le type de shadow auquel vous souhaitez faire référence, remplacez *thingName*, et *shadowName* le cas échéant, par leurs valeurs correspondantes, puis ajoutez cela au stub de rubrique comme indiqué dans le tableau suivant. N'oubliez pas que les rubriques sont sensibles à la casse.

Veuillez consulter Rubriques de shadow pour de plus amples informations sur les rubriques réservées pour les shadows.

Initialisation de l'appareil lors de la première connexion à AWS IoT

Une fois qu'un appareil s'est enregistré AWS IoT, il doit s'abonner à ces messages MQTT pour les ombres qu'il supporte.

Rubrique	Signification	Action qu'un appareil doit effectuer lors de la réception de cette rubrique
ShadowTopicPrefix `/delete/accepted`	La `delete` demande a été acceptée et l'ombre a été AWS IoT supprimée.	Actions nécessaires pour accompagner la suppression du shadow, telles que l'arrêt

Rubrique	Signification	Action qu'un appareil doit effectuer lors de la réception de cette rubrique
		de la publication des mises à jour.
ShadowTopicPrefix / delete/rejected	La delete demande a été rejetée par AWS IoT et l'ombre n'a pas été supprimée. Le corps du message contient les informations d'erreur.	Répondre au message d'erreur dans le corps du message.
ShadowTopicPrefix / get/accepted	La get demande a été acceptée par AWS IoT, et le corps du message contient le document fantôme actuel.	Actions nécessaires pour traiter le document d'état dans le corps du message.
ShadowTopicPrefix / get/rejected	La get demande a été rejetée par AWS IoT, et le corps du message contient les informations d'erreur.	Répondre au message d'erreur dans le corps du message.
ShadowTopicPrefix / update/accepted	La update demande a été acceptée par AWS IoT, et le corps du message contient le document fantôme actuel.	Confirmer que les données mises à jour dans le corps du message correspondent à l'état de l'appareil.
ShadowTopicPrefix / update/rejected	La update demande a été rejetée par AWS IoT, et le corps du message contient les informations d'erreur.	Répondre au message d'erreur dans le corps du message.

Rubrique	Signification	Action qu'un appareil doit effectuer lors de la réception de cette rubrique
ShadowTopicPrefix /update/delta	Le document fantôme a été mis à jour à la suite d'une demande adressée à AWS IoT, et le corps du message contient les modifications demandées.	Mettre à jour l'état de l'appareil pour qu'il corresponde à l'état souhaité dans le corps du message.
ShadowTopicPrefix /update/documents	Une mise à jour du shadow a été récemment réalisée et le corps du message contient le document shadow actuel.	Confirmer que l'état mis à jour dans le corps du message correspond à l'état de l'appareil.

Après s'être abonné aux messages du tableau précédent pour chaque shadow, l'appareil doit tester si les shadows qu'il prend en charge ont déjà été créés en publiant une rubrique `/get` sur chaque shadow. Si un message `/get/accepted` est reçu, le corps du message contient le document shadow que l'appareil peut utiliser pour initialiser son état. Si un message `/get/rejected` est reçu, le shadow doit être créé en publiant un message `/update` avec l'état actuel de l'appareil.

Par exemple, supposons que vous ayez un objet `My_IoT_Thing` qui n'ait pas d'ombres classiques ou nommées. Si vous publiez maintenant une `/get` demande sur le sujet réservé`$aws/things/My_IoT_Thing/shadow/get`, elle renvoie une erreur sur le `$aws/things/My_IoT_Thing/shadow/get/rejected` sujet car le sujet ne comporte aucune ombre. Pour résoudre cette erreur, publiez d'abord un `/update` message en utilisant la `$aws/things/My_IoT_Thing/shadow/update` rubrique présentant l'état actuel de l'appareil, par exemple la charge utile suivante.

```
{
 "state": {
  "reported": {
   "welcome": "aws-iot",
   "color": "yellow"
  }
 }
}
```

Une ombre classique est désormais créée pour l'objet et le message est publié dans le `$aws/things/My_IoT_Thing/shadow/update/accepted` sujet. Si vous publiez dans le sujet`$aws/things/My_IoT_Thing/shadow/get`, il renvoie une réponse au `$aws/things/My_IoT_Thing/shadow/get/accepted` sujet avec l'état de l'appareil.

Pour les ombres nommées, vous devez d'abord créer l'ombre nommée ou publier une mise à jour avec le nom de l'ombre avant d'utiliser la requête get. Par exemple, pour créer une ombre nommée `namedShadow1`, publiez d'abord les informations sur l'état de l'appareil dans la rubrique `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/update`. Pour récupérer les informations d'état, utilisez la `/get` requête pour l'ombre nommée, `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/get`.

Traitement des messages lorsque l'appareil est connecté à AWS IoT

Lorsqu'un appareil est connecté à AWS IoT, il peut recevoir des messages /update/delta et doit maintenir l'état de l'appareil adapté aux modifications survenues dans l'ombre en :

1. Lisant tous les messages /update/delta reçus et en synchronisant l'état de l'appareil pour qu'il y corresponde.
2. Publiant un message /update avec un corps de message `reported` doté de l'état actuel de l'appareil, chaque fois que l'état de l'appareil change.

Quand un appareil est connecté, il doit publier ces messages lorsque cela est indiqué.

Indication	Rubrique	Charge utile
L'état de l'appareil a changé.	*ShadowTopicPrefix* `/update`	Un document shadow avec la propriété `reported`.
L'appareil peut ne pas être synchronisé avec le shadow.	*ShadowTopicPrefix* `/get`	(empty)
Une action sur le dispositif indique qu'une ombre ne sera plus prise en charge par le dispositif, par exemple lorsque le dispositif est retiré ou remplacé.	*ShadowTopicPrefix* `/delete`	(empty)

Traitement des messages lorsque l'appareil se reconnecte à AWS IoT

Lorsqu'un appareil comportant une ou plusieurs ombres se connecte AWS IoT, il doit synchroniser son état avec celui de toutes les ombres qu'il supporte en :

1. Lisant tous les messages /update/delta reçus et en synchronisant l'état de l'appareil pour qu'il y corresponde.
2. Publiant un message /update avec un corps de message `reported` doté de l'état actuel de l'appareil.

Utilisation des shadows dans les applications et les services

Cette section décrit comment une application ou un service interagit avec le service AWS IoT Device Shadow. Cet exemple suppose que l'application ou le service interagit uniquement avec le shadow et, via le shadow, avec l'appareil. Cet exemple n'inclut aucune action de gestion, telle que la création ou la suppression de shadows.

Cet exemple utilise l'API REST du service AWS IoT Device Shadow pour interagir avec les ombres. Contrairement à l'exemple utilisé dansUtilisation des shadows sur les appareils, qui utilise un modèle de publish/subscribe communications model, this example uses the request/response communication de l'API REST. Cela signifie que l'application ou le service doit faire une demande avant de pouvoir recevoir une réponse de AWS IoT. Un inconvénient de ce modèle, toutefois, est qu'il ne prend pas en charge les notifications. Si votre application ou service nécessite des notifications rapides en cas de modification de l'état de l'appareil, envisagez les protocoles MQTT ou MQTT over WSS, qui prennent en charge le modèle de publish/subscribe communication, comme décrit dans. Utilisation des shadows sur les appareils

> **Important**
>
> Assurez-vous que l'utilisation des shadows par votre application ou service est cohérente et prise en charge par les implémentations correspondantes sur vos appareils. Prenez en compte, par exemple, la façon dont les shadows sont créés, mis à jour et supprimés, et la manière dont les mises à jour sont gérées sur l'appareil et dans les applications ou services qui accèdent au shadow. Votre conception doit indiquer clairement comment l'état de l'appareil est mis à jour et rapporté, et comment vos applications et services interagissent avec l'appareil et ses shadows.

L'URL de l'API REST pour un shadow nommé est :

```
https://endpoint/things/thingName/shadow?name=shadowName
```

et pour un shadow non nommé :

```
https://endpoint/things/thingName/shadow
```

où :

point de terminaison

Point de terminaison renvoyé par la commande CLI :

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

thingName

Nom de l'objet d'objet auquel le shadow appartient

shadowName

Nom du shadow nommé. Ce paramètre n'est pas utilisé avec des shadows non nommés.

Initialisation de l'application ou du service lors de la connexion à AWS IoT

Lorsque l'application se connecte pour la première fois à AWS IoT, elle doit envoyer une requête HTTP GET aux URLs ombres qu'elle utilise pour obtenir l'état actuel des ombres qu'elle utilise. Cela lui permet de synchroniser l'application ou le service avec le shadow.

L'état de traitement change lorsque l'application ou le service est connecté à AWS IoT

Lorsque l'application ou le service est connecté AWS IoT, il peut demander périodiquement l'état actuel en envoyant une requête HTTP GET sur URLs les ombres qu'il utilise.

Lorsqu'un utilisateur final interagit avec l'application ou le service pour modifier l'état de l'appareil, l'application ou le service peut envoyer une requête HTTP POST aux URLs ombres qu'il utilise pour mettre à jour l'`desired`état de l'ombre. Cette demande renvoie la modification qui a été acceptée,

mais vous devrez peut-être interroger le shadow en effectuant des demandes HTTP GET jusqu'à ce que l'appareil ait mis à jour le shadow avec son nouvel état.

Détection d'un appareil connecté

Pour déterminer si un appareil est actuellement connecté, incluez une propriété `connected` dans le document shadow et utilisez un message MQTT Last Will and Testament (LWT) pour définir la propriété `connected` sur `false` si un appareil est déconnecté en raison d'une erreur.

> **Note**
>
> Les messages MQTT LWT envoyés à des sujets AWS IoT réservés (sujets commençant par $) sont ignorés par le service AWS IoT Device Shadow. Cependant, ils sont traités par les clients abonnés et par le moteur de AWS IoT règles. Vous devrez donc créer un message LWT envoyé à un sujet non réservé et une règle qui republie le message MQTT LWT sous forme de message de mise à jour parallèle au sujet de mise à jour réservé du shadow,. *`ShadowTopicPrefix`*`/update`

Pour envoyer un message LWT au service Device Shadow

1. Créez une règle qui republie le message MQTT LWT sur la rubrique réservée. L'exemple suivant est une règle à l'écoute d'un message sur la `my/things/myLightBulb/update`rubrique et qui le republie dans `$aws/things/myLightBulb/shadow/update`.

```
{
    "rule": {
    "ruleDisabled": false,
    "sql": "SELECT * FROM 'my/things/myLightBulb/update'",
    "description": "Turn my/things/ into $aws/things/",
    "actions": [
        {
        "republish": {
            "topic": "$$aws/things/myLightBulb/shadow/update",
            "roleArn": "arn:aws:iam:123456789012:role/aws_iot_republish"
            }
        }
     ]
   }
}
```

2. Lorsque l'appareil se connecte à AWS IoT, il enregistre un message LWT dans un sujet non réservé afin que la règle de republication le reconnaisse. Dans cet exemple, cette rubrique est `my/things/myLightBulb/update` et elle définit la propriété connectée sur `false`.

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

3. Après la connexion, l'appareil publie un message sur sa rubrique de mise à jour de shadow, `$aws/things/myLightBulb/shadow/update`, pour rapporter son état actuel, ce qui inclut la définition de sa propriété `connected` sur `true`.

```
{
     "state": {
        "reported": {
            "connected":"true"
        }
    }
}
```

4. Avant que l'appareil ne se déconnecte gracieusement, il publie un message sur sa rubrique de mise à jour de shadow, `$aws/things/myLightBulb/shadow/update`, pour rapporter son état le plus récent, ce qui inclut la définition de sa propriété `connected` sur `false`.

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

5. Si le périphérique se déconnecte en raison d'une erreur, le courtier de messages publie le AWS IoT message LWT de l'appareil au nom de celui-ci. La règle de republication détecte ce message et publie le message de mise à jour de shadow pour mettre à jour la propriété `connected` du shadow de l'appareil.

Simulation des communications du service Device Shadow

Cette rubrique indique comment le service Device Shadow agit en tant qu'intermédiaire et permet aux appareils et aux applications d'utiliser un shadow pour mettre à jour, stocker et récupérer l'état d'un appareil.

Pour démontrer l'interaction décrite dans cette rubrique et pour l'explorer plus en détail, vous aurez besoin d'un Compte AWS et d'un système sur lesquels vous pourrez exécuter le AWS CLI. Si vous ne les avez pas, vous pouvez toujours voir l'interaction dans les exemples de code.

Dans cet exemple, la AWS IoT console représente le périphérique. Le AWS CLI représente l'application ou le service qui accède à l'appareil par le biais de l'ombre. L' AWS CLI interface est très similaire à l'API avec laquelle une application peut communiquer AWS IoT. L'appareil dans cet exemple est une ampoule intelligente et l'application affiche l'état de cette ampoule et peut le modifier.

Configuration de la simulation

Ces procédures initialisent la simulation en ouvrant la console AWS IoT, qui simule votre appareil, et la fenêtre de ligne de commande qui simule votre application.

Pour configurer votre environnement de simulation

1. Vous aurez besoin d'un Compte AWS pour exécuter vous-même les exemples de cette rubrique. Si vous n'en avez pas Compte AWS, créez-en un, comme décrit dansConfigurez Compte AWS.
2. Ouvrez la console AWS IoT et, dans le menu de gauche, choisissez Test pour ouvrir le client MQTT.
3. Dans une autre fenêtre, ouvrez une fenêtre de terminal sur un système où l'interface AWS CLI est installée.

Deux fenêtres doivent être ouvertes : l'une avec la AWS IoT console sur la page Test et l'autre avec une invite de ligne de commande.

Initialisation de l'appareil

Dans cette simulation, nous allons travailler avec un objet nommé mySimulatedThing, et son ombre nommée SimShadow1.

Create thing Object et sa politique en matière d'IoT

Pour créer un objet, dans la AWS IoT console :

1. Sélectionnez Gérer, puis choisissez votre objet .
2. Cliquez sur le bouton Créer si des éléments sont répertoriés, sinon cliquez sur Enregistrer un seul objet pour créer un seul AWS IoT élément.
3. Entrez le nom`mySimulatedThing`, laissez les autres paramètres par défaut, puis cliquez sur Suivant.
4. Utilisez la création de certificats en un clic pour générer les certificats qui authentifieront la connexion de l'appareil à AWS IoT. Cliquez sur Activer pour activer le certificat.
5. Vous pouvez joindre la politique `My_IoT_Policy` qui autoriserait l'appareil à publier et à s'abonner aux sujets réservés au MQTT. Pour des étapes plus détaillées sur la création d'un AWS IoT objet et sur la création de cette politique, consultezCréez un objet.

Crée une ombre nommée pour l'objet

Vous pouvez créer une ombre nommée pour un objet en publiant une demande de mise à jour dans la rubrique `$aws/things/mySimulatedThing/shadow/name/simShadow1/update`, comme décrit ci-dessous.

Ou, pour créer une ombre nommée :

1. Dans la AWS IoT console, choisissez votre objet dans la liste des objets affichés, puis choisissez Shadows.
2. Choisissez Ajouter une ombre, entrez le nom`simShadow1`, puis choisissez Créer pour ajouter l'ombre nommée.

S'abonner et publier sur des sujets MQTT réservés

Dans la console, souscrivez aux sujets MQTT réservés. Ces rubriques sont les réponses aux actions `get`, `update` et `delete`, afin que votre appareil soit prêt à recevoir les réponses après avoir publié une action.

Pour vous abonner à une rubrique MQTT dans le client MQTT

1. Dans le client MQTT, choisissez Publier dans une rubrique.

2. Entrez le `get`, `update`, et les `delete` sujets auxquels vous souhaitez vous abonner. Copiez un sujet à la fois dans la liste suivante, collez-le dans le champ Filtre de sujet, puis cliquez sur S'abonner. Vous devriez voir les sujets apparaître sous Abonnements.

   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta`
   - `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/documents`

   À ce stade, votre appareil simulé est prêt à recevoir les rubriques telles qu'elles sont publiées par AWS IoT.

Pour vous abonner à une rubrique MQTT dans le client MQTT

Une fois qu'un appareil s'est initialisé et s'est abonné aux rubriques de réponse, il doit exécuter une requête portant sur les shadows qu'il prend en charge. Cette simulation ne prend en charge qu'une seule ombre, l'ombre qui supporte un objet nommé SimShadow1. mySimulatedThing

Pour obtenir l'état actuel du shadow à partir du client MQTT

1. Dans le client MQTT, choisissez Publier dans une rubrique.
2. Sous Publish, entrez l’objet suivant et supprimez tout contenu de la fenêtre de corps de message ci-dessous où vous avez entré la rubrique à obtenir. Vous pouvez ensuite choisir Publier dans le sujet pour publier la demande. `$aws/things/mySimulatedThing/shadow/name/simShadow1/get`.

   Si vous n'avez pas créé l'ombre nommée, `simShadow1`, vous recevez un message dans `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected` l’objet et le `code` est `404`, comme dans cet exemple où l'ombre n'a pas été créée, nous allons donc la créer ensuite.

```
{
```

```
  "code": 404,
  "message": "No shadow exists with name: 'simShadow1'"
}
```

Pour créer un shadow avec l'état actuel de l'appareil

1. Dans le client MQTT, choisissez Publier dans une rubrique et entrez l’objet.

   ```
   $aws/things/mySimulatedThing/shadow/name/simShadow1/update
   ```

2. Dans la fenêtre de corps de message ci-dessous où vous avez entré la rubrique, entrez ce document shadow pour montrer que l'appareil rapporte son ID et sa couleur actuelle en valeurs RVB. Choisissez Publier pour publier la demande.

   ```
   {
     "state": {
       "reported": {
         "ID": "SmartLamp21",
         "ColorRGB": [
           128,
           128,
           128
         ]
       }
     },
     "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
   }
   ```

Si vous recevez un message dans le sujet :

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted` : Cela signifie que l'ombre a été créée et que le corps du message contient le document fantôme actuel.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected` : Vérifiez l'erreur dans le corps du message.
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted` : L'ombre existe déjà et le corps du message contient l'état actuel de l'ombre, comme dans cet exemple. Avec cela, vous pouvez définir votre appareil ou confirmer qu'il correspond à l'état du shadow.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
  "version": 3,
  "timestamp": 1591140517,
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```


Envoi d'une mise à jour à partir de l'application

Cette section utilise le AWS CLI pour montrer comment une application peut interagir avec une ombre.

Pour obtenir l'état actuel de l'ombre à l'aide du AWS CLI

Sur la ligne de commande, entrez la commande ci-dessous.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /
dev/stdout
```

Sur les plateformes Windows, vous pouvez utiliser `con` à la place de`/dev/stdout`.

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1
 con
```

Comme le shadow existe et a été initialisé par l'appareil pour refléter son état actuel, il doit renvoyer le document shadow suivant.

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
```

```
  "version": 3,
  "timestamp": 1591141111
}
```

L'application peut utiliser cette réponse pour initialiser sa représentation de l'état de l'appareil.

Si l'application met à jour l'état, par exemple lorsqu'un utilisateur final change la couleur de notre ampoule intelligente en jaune, l'application enverra une commande update-thing-shadow. Cette commande correspond à l'API REST `UpdateThingShadow`.

Pour mettre à jour un shadow à partir d'une application

Sur la ligne de commande, entrez la commande ci-dessous.

AWS CLI v2.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
    --cli-binary-format raw-in-base64-out \
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

AWS CLI v1.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name
 simShadow1 \
    --payload '{"state":{"desired":{"ColorRGB":
[255,255,0]}},"clientToken":"21b21b21-bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

Si elle réussit, cette commande doit renvoyer le document shadow suivant.

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
```

```
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    }
  },
  "version": 4,
  "timestamp": 1591141596,
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

Réponse à une mise à jour sur l'appareil

Pour en revenir au client MQTT dans la AWS console, vous devriez voir les messages AWS IoT publiés pour refléter la commande de mise à jour émise dans la section précédente.

Pour afficher les messages de mise à jour dans le client MQTT

Dans le client MQTT, choisissez $ aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta dans la colonne Abonnements. Si le nom de la rubrique est tronqué, vous pouvez faire une pause dessus pour voir le nom complet. Dans le journal des rubriques de cette rubrique, vous devriez voir un `/delta` message similaire à celui-ci.

```
{
  "version": 4,
  "timestamp": 1591141596,
  "state": {
    "ColorRGB": [
      255,
      255,
      0
    ]
```

```
  },
  "metadata": {
    "ColorRGB": [
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      }
    ]
  },
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

Votre appareil traite le contenu de ce message pour définir l'état de l'appareil, afin qu'il corresponde à l'état `desired` dans le message.

Une fois que l'appareil a mis à jour l'état pour qu'il corresponde à l'`desired`état indiqué dans le message, il doit renvoyer le nouvel état signalé AWS IoT en publiant un message de mise à jour. Cette procédure simule cela dans le client MQTT.

Pour mettre à jour le shadow à partir de l'appareil

1. Dans le client MQTT, choisissez Publier dans une rubrique.
2. Dans la fenêtre du corps du message, dans le champ de sujet situé au-dessus de la fenêtre du corps du message, entrez le sujet de l'ombre suivi de l'`/update`action : `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` et dans le corps du message, entrez ce document fantôme mis à jour, qui décrit l'état actuel de l'appareil. Choisissez Publier pour publier l'état de l'appareil mis à jour.

   ```
   {
     "state": {
       "reported": {
         "ColorRGB": [255,255,0]
         }
     },
     "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
   }
   ```

Si le message a bien été reçu par AWS IoT, vous devriez voir une nouvelle réponse dans le journal des aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted messages $ du client MQTT avec l'état actuel de l'ombre, comme dans cet exemple.

```
{
  "state": {
    "reported": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "reported": {
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
      ]
    }
  },
  "version": 5,
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Une mise à jour réussie de l'état signalé du périphérique AWS IoT entraîne également l'envoi d'une description complète de l'état d'ombre dans un message adressé au `update/documents` sujet, tel que le corps du message résultant de la mise à jour instantanée effectuée par le périphérique dans la procédure précédente.

```
{
```

```
  "previous": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          128,
          128,
          128
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          },
          {
```

```
            "timestamp": 1591140517
          }
        ]
      }
    },
    "version": 4
  },
  "current": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          255,
          255,
          0
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
```

```
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          }
        ]
      }
    },
    "version": 5
  },
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

Observation de la mise à jour dans l'application

L'application peut désormais interroger le shadow pour obtenir l'état actuel, tel qu'il a été rapporté par l'appareil.

Pour obtenir l'état actuel de l'ombre à l'aide du AWS CLI

1. Sur la ligne de commande, entrez la commande ci-dessous.

   ```
   aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
    simShadow1 /dev/stdout
   ```

   Sur les plateformes Windows, vous pouvez utiliser à la `con` place de`/dev/stdout`.

   ```
   aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
    simShadow1 con
   ```

2. Comme le shadow vient d'être mis à jour par l'appareil pour refléter son état actuel, il doit renvoyer le document shadow suivant.

   ```
   {
     "state": {
       "desired": {
         "ColorRGB": [
   ```

```
        255,
        255,
        0
      ]
    },
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    },
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
      ]
    }
```

```
  },
  "version": 5,
  "timestamp": 1591143269
}
```

Au-delà de la simulation

Expérimentez avec l'interaction entre l'interface AWS CLI (représentant l'application) et la console (représentant l'appareil) pour modéliser votre solution IoT.

Interaction avec les shadows

Cette rubrique décrit les messages associés à chacune des trois méthodes fournies par AWS IoT qui permettent de travailler avec les shadows. Il s'agit notamment des méthodes suivantes :

`UPDATE`

Crée un shadow s'il n'existe pas ou met à jour le contenu d'un shadow existant avec les informations d'état fournies dans le corps de message. AWS IoT enregistre un horodatage avec chaque mise à jour pour indiquer quand l'état a été mis à jour pour la dernière fois. Lorsque l'état de l'ombre change, AWS IoT envoie `/delta` des messages à tous les abonnés MQTT avec la différence entre les `reported` états `desired` et. Les appareils ou les applications qui reçoivent un message `/delta` peuvent effectuer des actions en fonction de cette différence. Par exemple, un appareil peut mettre à jour son état à l'état souhaité, ou une application peut mettre à jour son interface utilisateur pour refléter le changement d'état de l'appareil.

`GET`

Récupère un document shadow actuel qui contient l'état complet du shadow, y compris les métadonnées.

`DELETE`

Supprime l'ombre de l'appareil et son contenu.

Vous ne pouvez pas restaurer un document fantôme supprimé sur un appareil, mais vous pouvez en créer un nouveau avec le nom d'un document fantôme supprimé sur un appareil. Si vous créez un document fantôme de terminal portant le même nom qu'un document supprimé au cours des dernières 48 heures, le numéro de version du nouveau document fantôme de terminal suivra celui du document supprimé. Si le document fantôme d'un appareil a été supprimé pendant plus de 48

heures, le numéro de version d'un nouveau document fantôme de l'appareil portant le même nom sera 0.

Support du protocole

AWS IoT prend en charge le MQTT et une API REST via les protocoles HTTPS pour interagir avec les ombres. AWS IoT fournit un ensemble de sujets de demande et de réponse réservés pour les actions de publication et d'abonnement MQTT. Les appareils et les applications doivent s'abonner aux sujets de réponse avant de publier dans un sujet de demande afin d'obtenir des informations sur le AWS IoT traitement de la demande. Pour plus d'informations, consultez Rubriques MQTT de Device Shadow et API REST Device Shadow.

Demande d'état et génération de rapport d'état

Lorsque vous concevez votre solution IoT à l'aide de AWS IoT et d'ombres, vous devez déterminer les applications ou les appareils qui demanderont des modifications et ceux qui les mettront en œuvre. Généralement, un appareil implémente les modifications et les rapporte au shadow, et les applications et services y répondent et demandent les modifications dans le shadow. Votre solution peut être différente, mais les exemples de cette rubrique supposent que l'application ou le service client demande les modifications dans le shadow et que l'appareil effectue ces modifications et les rapporte en retour au shadow.

Mise à jour d'un shadow

Votre application ou service peut mettre à jour l'état d'un shadow à l'aide de l'API UpdateThingShadow ou en publiant dans la rubrique /update. Les mises à jour concernent uniquement les champs spécifiés dans la demande.

Mise à jour d'un shadow lorsqu'un client demande un changement d'état

Quand un client demande un changement d'état dans un shadow à l'aide du protocole MQTT

1. Le client doit disposer d'un document shadow actuel pour pouvoir identifier les propriétés à modifier. Veuillez consulter l'action /get pour voir comment obtenir le document shadow actuel.
2. Le client s'abonne aux rubriques MQTT suivantes :

   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`

- `$aws/things/thingName/shadow/name/shadowName/update/delta`
- `$aws/things/thingName/shadow/name/shadowName/update/documents`

3. Le client publie une rubrique de demande `$aws/things/thingName/shadow/name/shadowName/update` avec un document d'état qui contient l'état souhaité du shadow. Seules les propriétés à modifier doivent être incluses dans ce document. Ceci est un exemple de document avec l'état souhaité.

```
{
  "state": {
    "desired": {
      "color": {
        "r": 10
      },
      "engine": "ON"
    }
  }
}
```

4. Si la demande de mise à jour est valide, AWS IoT met à jour l'état souhaité dans l'ombre et publie des messages sur les sujets suivants :

   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`

   Le message `/update/accepted` contient un document shadow /document d'état de la réponse accepté et le message `/update/delta` contient un document shadow /documents d'état de la réponse delta.

5. Si la demande de mise à jour n'est pas valide, AWS IoT publie un message avec le `$aws/things/thingName/shadow/name/shadowName/update/rejected` sujet avec un document Document de réponse d'erreur fantôme décrivant l'erreur.

Quand un client demande un changement d'état dans un shadow à l'aide de l'API

1. Le client appelle l'API `UpdateThingShadow` avec un document d'état Document d'état de demande comme corps de message.

2. Si la demande était valide, AWS IoT renvoie un code de réponse de réussite HTTP et un document /document d'état de la réponse accepté fantôme dans le corps du message de réponse.

   AWS IoT publiera également un message MQTT sur le `$aws/things/thingName/shadow/name/shadowName/update/delta` sujet avec un document /documents d'état de la réponse delta fantôme pour tous les appareils ou clients qui s'y abonnent.
3. Si la demande n'est pas valide, AWS IoT renvoie un code de réponse d'erreur HTTP an Document de réponse d'erreur comme corps du message de réponse.

Lorsque l'appareil reçoit l'état `/desired` sur la rubrique `/update/delta`, il effectue les modifications souhaitées sur l'appareil. Il envoie ensuite un message à la rubrique `/update` pour rapporter son état actuel au shadow.

Mise à jour d'un shadow lorsqu'un appareil rapporte son état actuel

Quand un appareil rapporte son état actuel au shadow à l'aide du protocole MQTT

1. L'appareil doit s'abonner aux rubriques MQTT suivantes avant de mettre à jour le shadow :

   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
   - `$aws/things/thingName/shadow/name/shadowName/update/delta`
   - `$aws/things/thingName/shadow/name/shadowName/update/documents`
2. L'appareil rapporte son état actuel en publiant un message dans la rubrique `$aws/things/thingName/shadow/name/shadowName/update` qui rapporte l'état actuel, comme dans cet exemple.

```
{
    "state": {
        "reported" : {
            "color" : { "r" : 10 },
            "engine" : "ON"
        }
    }
}
```

3. S'il AWS IoT accepte la mise à jour, il publie un message aux `$aws/things/thingName/shadow/name/shadowName/update/accepted` rubriques avec un document /document d'état de la réponse accepté fantôme.
4. Si la demande de mise à jour n'est pas valide, AWS IoT publie un message avec le `$aws/things/thingName/shadow/name/shadowName/update/rejected` sujet avec un document Document de réponse d'erreur fantôme décrivant l'erreur.

Quand un appareil rapporte son état actuel au shadow à l'aide de l'API

1. L'appareil appelle l'API `UpdateThingShadow` avec un document d'état Document d'état de demande comme corps de message.
2. Si la demande était valide, AWS IoT met à jour le fantôme et renvoie un code de réponse HTTP avec un document /document d'état de la réponse accepté fantôme comme corps de message de réponse.

   AWS IoT publiera également un message MQTT sur le `$aws/things/thingName/shadow/name/shadowName/update/delta` sujet avec un document /documents d'état de la réponse delta fantôme pour tous les appareils ou clients qui s'y abonnent.
3. Si la demande n'est pas valide, AWS IoT renvoie un code de réponse d'erreur HTTP an Document de réponse d'erreur comme corps du message de réponse.

Verrouillage optimiste

Vous pouvez utiliser la version du document d'état pour vous assurer que vous mettez à jour la version la plus récente d'un document shadow d'appareil. Lorsque vous fournissez une version dans une demande de mise à jour, le service rejette la demande avec un code de réponse de conflit HTTP 409 si la version actuelle du document d'état ne correspond pas à la version fournie. Le code de réponse au conflit peut également apparaître sur n'importe quelle API modifiée`ThingShadow`, notamment`DeleteThingShadow`.

Par exemple :

Document initial :

```
{
  "state": {
    "desired": {
      "colors": [
```

```
        "RED",
        "GREEN",
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

Mise à jour : (la version ne correspond pas ; cette demande est rejetée)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 9
}
```

Résultat:

```
{
  "code": 409,
  "message": "Version conflict",
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

Mise à jour : (la version correspond ; cette demande est acceptée)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

État final :

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 11
}
```

Récupération d'un document Shadow

Vous pouvez récupérer un document shadow à l'aide de l'API GetThingShadow ou en vous abonnant et en publiant dans la rubrique /get. Ceci récupère un document shadow complet, y compris tout delta entre les états `desired` et `reported`. La procédure pour cette tâche est la même que l'appareil ou un client effectue la demande.

Pour récupérer un document shadow à l'aide du protocole MQTT

1. L'appareil ou le client doit s'abonner à ces rubriques MQTT avant de mettre à jour le shadow :
   - `$aws/things/thingName/shadow/name/shadowName/get/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/get/rejected`
2. L'appareil ou le client publie un message dans la rubrique `$aws/things/thingName/shadow/name/shadowName/get` avec un corps de message vide.
3. Si la demande aboutit, AWS IoT publie un message dans le `$aws/things/thingName/shadow/name/shadowName/get/accepted` sujet avec un /document d'état de la réponse accepté dans le corps du message.
4. Si la demande n'est pas valide, AWS IoT publie un message dans le `$aws/things/thingName/shadow/name/shadowName/get/rejected` sujet avec un Document de réponse d'erreur dans le corps du message.

Pour récupérer un document shadow à l'aide d'une API REST

1. L'appareil ou le client appelle l'API `GetThingShadow` avec un corps de message vide.

2. Si la demande est valide, AWS IoT renvoie un code de réponse HTTP avec un document / document d'état de la réponse accepté fantôme comme corps du message de réponse.
3. Si la demande n'est pas valide, AWS IoT renvoie un code de réponse d'erreur HTTP an Document de réponse d'erreur comme corps du message de réponse.

Suppression de données shadow

Il existe deux façons de supprimer des données shadow : vous pouvez supprimer les propriétés spécifiques dans le document shadow et vous pouvez supprimer complètement le shadow.

- Pour supprimer des propriétés spécifiques d'un shadow, mettez à jour le shadow. Toutefois, définissez la valeur des propriétés à supprimer sur `null`. Les champs dotés d'une valeur `null` sont supprimés du document shadow.
- Pour supprimer le shadow entier, utilisez l'API DeleteThingShadow ou publiez dans la rubrique / delete.

> **Note**
>
> La suppression d'une ombre ne remet pas immédiatement son numéro de version à zéro. Il sera remis à zéro au bout de 48 heures.

Suppression d'une propriété dans un document shadow

Pour supprimer une propriété dans un shadow à l'aide du protocole MQTT

1. L'appareil ou le client doit disposer d'un document shadow actuel pour pouvoir identifier les propriétés à modifier. Veuillez consulter Récupération d'un document Shadow pour obtenir des informations sur la façon d'obtenir le document shadow actuel.
2. L'appareil ou le client s'abonne aux rubriques MQTT suivantes :
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
3. L'appareil ou le client publie une rubrique de demande `$aws/things/thingName/shadow/name/shadowName/update` avec un document d'état qui attribue des valeurs `null` aux

propriétés du shadow à supprimer. Seules les propriétés à modifier doivent être incluses dans ce document. Voici un exemple de document qui supprime la propriété `engine`.

```
{
  "state": {
    "desired": {
      "engine": null
    }
  }
}
```

4. Si la demande de mise à jour est valide, AWS IoT supprime les propriétés spécifiées dans l'ombre et publie un message avec le `$aws/things/thingName/shadow/name/shadowName/update/accepted` sujet avec un document /document d'état de la réponse accepté fantôme dans le corps du message.
5. Si la demande de mise à jour n'est pas valide, AWS IoT publie un message avec le `$aws/things/thingName/shadow/name/shadowName/update/rejected` sujet avec un document Document de réponse d'erreur fantôme décrivant l'erreur.

Pour supprimer une propriété d'un shadow à l'aide de l'API REST

1. L'appareil ou le client appelle l'API `UpdateThingShadow` avec un Document d'état de demande qui attribue des valeurs `null` aux propriétés du shadow à supprimer. Incluez uniquement les propriétés que vous souhaitez supprimer dans le document. Voici un exemple de document qui supprime la propriété `engine`.

```
{
  "state": {
    "desired": {
      "engine": null
    }
  }
}
```

2. Si la demande était valide, AWS IoT renvoie un code de réponse de réussite HTTP et un document /document d'état de la réponse accepté fantôme dans le corps du message de réponse.
3. Si la demande n'est pas valide, AWS IoT renvoie un code de réponse d'erreur HTTP an Document de réponse d'erreur comme corps du message de réponse.

Suppression d'un shadow

Vous trouverez ci-après quelques considérations relatives à la suppression de l'ombre d'un appareil.

- La définition de l'état de shadow de l'appareil sur `null` ne supprime pas le shadow. La version de shadow sera incrémentée lors de la prochaine mise à jour.
- La suppression d'un shadow d'appareil ne supprime pas l'objet d'objet. La suppression d'un objet d'objet ne supprime pas le shadow d'appareil correspondant.
- La suppression d'une ombre ne remet pas immédiatement son numéro de version à zéro. Il sera remis à zéro au bout de 48 heures.

Pour supprimer un shadow à l'aide du protocole MQTT

1. L'appareil ou le client s'abonne aux rubriques MQTT suivantes :
   - `$aws/things/`*`thingName`*`/shadow/name/`*`shadowName`*`/delete/accepted`
   - `$aws/things/`*`thingName`*`/shadow/name/`*`shadowName`*`/delete/rejected`
2. L'appareil ou le client publie un `$aws/things/`*`thingName`*`/shadow/name/`*`shadowName`*`/delete` avec un tampon de messages vide.
3. Si la demande de suppression est valide, AWS IoT supprime l'ombre et publie un message avec le `$aws/things/`*`thingName`*`/shadow/name/`*`shadowName`*`/delete/accepted` sujet et un document /document d'état de la réponse accepté fantôme abrégé dans le corps du message. Voici un exemple du message de suppression accepté :

   ```
   {
     "version": 4,
     "timestamp": 1591057529
   }
   ```

4. Si la demande de mise à jour n'est pas valide, AWS IoT publie un message avec le `$aws/things/`*`thingName`*`/shadow/name/`*`shadowName`*`/delete/rejected` sujet avec un document Document de réponse d'erreur fantôme décrivant l'erreur.

Pour supprimer un shadow à l'aide de l'API REST

1. L'appareil ou le client appelle l'API `DeleteThingShadow` avec un tampon de messages vide.

2. Si la demande était valide, AWS IoT renvoie un code de réponse de réussite HTTP /document d'état de la réponse accepté et un document /document d'état de la réponse accepté fantôme abrégé dans le corps du message. Voici un exemple du message de suppression accepté :

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

3. Si la demande n'est pas valide, AWS IoT renvoie un code de réponse d'erreur HTTP an Document de réponse d'erreur comme corps du message de réponse.

API REST Device Shadow

Un shadow expose l'URI suivante pour mettre à jour les informations d'état :

```
https://account-specific-prefix-ats.iot.region.amazonaws.com/things/thingName/shadow
```

Le point de terminaison est spécifique à votre Compte AWS. Pour trouver votre point de terminaison, vous pouvez :

- Utilisez la commande describe-endpoint du AWS CLI.
- Utilisez les paramètres AWS IoT de la console. Dans Paramètres, le point de terminaison est répertorié sous Point de terminaison personnalisé
- Utilisez la page de détails des éléments de la AWS IoT console. Dans la console :
  1. Ouvrez Gérer et sous Gérer, sélectionnez Objets.
  2. Dans la liste des éléments, choisissez l'élément pour lequel vous souhaitez obtenir l'URI du point de terminaison.
  3. Choisissez l'onglet Device Shadows et choisissez votre ombre. Vous pouvez consulter l'URI du point de terminaison dans la section Device Shadow URL de la page de détails du Device Shadow.

Le format du point de terminaison est le suivant :

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

L'API Shadow REST suit les mêmes protocols/port mappages HTTPS que ceux décrits dansProtocoles de communication des appareils.

> **Note**
>
> Pour utiliser le APIs, vous devez l'utiliser `iotdevicegateway` comme nom de service pour l'authentification. Pour plus d'informations, consultez Io TData Plane.

Actions d'API

- GetThingShadow
- UpdateThingShadow
- DeleteThingShadow
- ListNamedShadowsForThing

Vous pouvez également utiliser l'API pour créer une ombre nommée en la fournissant dans le `name=`*`shadowName`* cadre du paramètre de requête de l'API.

GetThingShadow

Obtient le shadow de l'objet spécifié.

Le document d'état de réponse comprend le delta entre les états `desired` et `reported`.

Demande

La demande comprend les en-têtes HTTP standard, plus l'URI suivante :

```
HTTP GET https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

Le paramètre de requête `name` n'est pas requis pour les shadows non nommés (classiques).

Réponse

En cas de réussite, la réponse comprend les en-têtes HTTP standard, plus le code et le corps suivants :

```
HTTP 200
```

```
Response Body: response state document
```

Pour plus d'informations, consultez Exemple de document d'état de réponse.

Autorisation

La récupération d'un shadow nécessite une stratégie qui permet au mandataire de réaliser l'action `iot:GetThingShadow`. Le service Device Shadow accepte deux formes d'authentification : Signature Version 4 avec des informations d'identification IAM ou authentification mutuelle TLS avec un certificat client.

Voici un exemple de stratégie qui permet à un mandataire de récupérer un shadow d'appareil :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:GetThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

UpdateThingShadow

Met à jour le shadow de l'objet spécifié.

Les mises à jour concernent uniquement les champs spécifiés dans le document d'état de la demande. Tout champ avec une valeur `null` est supprimé du shadow d'appareil.

Demande

La demande comprend les en-têtes HTTP standard, plus l'URI et le corps suivants :

```
HTTP POST https://endpoint/things/thingName/shadow?name=shadowName
Request body: request state document
```

Le paramètre de requête `name` n'est pas requis pour les shadows non nommés (classiques).

Pour plus d'informations, consultez Exemple de document d'état de la demande.

Réponse

En cas de réussite, la réponse comprend les en-têtes HTTP standard, plus le code et le corps suivants :

```
HTTP 200
Response body: response state document
```

Pour plus d'informations, consultez Exemple de document d'état de réponse.

Autorisation

La mise à jour d'un shadow nécessite une stratégie qui permet au mandataire de réaliser l'action `iot:UpdateThingShadow`. Le service Device Shadow accepte deux formes d'authentification : Signature Version 4 avec des informations d'identification IAM ou authentification mutuelle TLS avec un certificat client.

Voici un exemple de stratégie qui permet à un mandataire de mettre à jour un shadow d'appareil :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:UpdateThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

DeleteThingShadow

Supprime le shadow de l'objet spécifié.

Demande

La demande comprend les en-têtes HTTP standard, plus l'URI suivante :

```
HTTP DELETE https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

Le paramètre de requête `name` n'est pas requis pour les shadows non nommés (classiques).

Réponse

En cas de réussite, la réponse comprend les en-têtes HTTP standard, plus le code et le corps suivants :

```
HTTP 200
Response body: Empty response state document
```

Notez que la suppression d'une ombre ne rétablit pas son numéro de version à 0.

Autorisation

La suppression d'un shadow d'appareil nécessite une stratégie qui permet au mandataire de réaliser l'action `iot:DeleteThingShadow`. Le service Device Shadow accepte deux formes d'authentification : Signature Version 4 avec des informations d'identification IAM ou authentification mutuelle TLS avec un certificat client.

Voici un exemple de stratégie qui permet à un mandataire de supprimer un shadow d'appareil :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:DeleteThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
```

```
  ]
}
```

ListNamedShadowsForThing

Répertorie les shadows de l'objet spécifié.

Demande

La demande comprend les en-têtes HTTP standard, plus l'URI suivante :

```
HTTP GET /api/things/shadow/ListNamedShadowsForThing/thingName?
nextToken=nextToken&pageSize=pageSize
Request body: (none)
```

nextToken

Jeton permettant de récupérer l'ensemble suivant de résultats.

Cette valeur est renvoyée sur les résultats paginés et est utilisée dans l'appel qui renvoie la page suivante.

pageSize

Nombre de noms de shadows à renvoyer dans chaque appel. Voir aussi `nextToken`.

thingName

Nom de l'objet pour lequel répertorier les shadows nommés.

Réponse

En cas de succès, la réponse comprend les en-têtes HTTP standard ainsi que le code de réponse suivant et un message de type Document de réponse de liste de noms de shadows

> **Note**
>
> Le shadow non nommé (classique) n'apparaît pas dans cette liste. La réponse est une liste vide si vous n'avez qu'une ombre classique ou si celle `thingName` que vous spécifiez n'existe pas.

```
HTTP 200
Response body: Shadow name list document
```

Autorisation

L'énumération de l'ombre d'un appareil nécessite une politique permettant à l'appelant d'effectuer l'action `iot:ListNamedShadowsForThing`. Le service Device Shadow accepte deux formes d'authentification : Signature Version 4 avec des informations d'identification IAM ou authentification mutuelle TLS avec un certificat client.

Voici un exemple de stratégie qui permet à un mandataire de répertorier les shadows nommés d'un objet :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:ListNamedShadowsForThing",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

Rubriques MQTT de Device Shadow

Le service Device Shadow utilise des rubriques MQTT réservées pour permettre à des appareils et à des applications d'obtenir, de mettre à jour ou de supprimer les informations d'état d'un appareil (shadow).

La publication et l'abonnement à des rubriques shadow nécessite une autorisation basée sur les rubriques. AWS IoT se réserve le droit d'ajouter de nouvelles rubriques à la structure de rubriques existante. C'est pourquoi nous vous recommandons d'éviter les abonnements de caractère générique aux rubriques shadow. Par exemple, évitez de vous abonner à des filtres de sujets, `$aws/things/`

`thingName/shadow/#` car le nombre de sujets correspondant à ce filtre de sujet peut augmenter à mesure que de nouveaux AWS IoT sujets secondaires seront introduits. Pour consulter des messages publiés dans ces rubriques, consultez Interaction avec les shadows.

Les shadows peuvent être nommés ou non (classique). Les rubriques utilisées par chacun d'eux ne diffèrent que par le préfixe de rubrique. Ce tableau indique le préfixe de rubrique utilisé par chaque type de shadow.

Valeur ***ShadowTopicPrefix***	Type de shadow
`$aws/things/` *`thingName`* `/shadow`	Shadow non nommé (classique)
`$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`*	Shadow nommé

Pour créer une rubrique complète, sélectionnez le *`ShadowTopicPrefix`* pour le type de shadow auquel vous souhaitez faire référence, remplacez *`thingName`*, et *`shadowName`* le cas échéant, par leurs valeurs correspondantes, puis ajoutez cela au stub de rubrique comme indiqué dans les sections suivantes.

Voici les rubriques MQTT utilisés pour interagir avec les shadows.

Rubriques

- /get
- /get/accepted
- /get/rejected
- /update
- /update/delta
- /update/accepted
- /update/documents
- /update/rejected
- /delete
- /delete/accepted
- /delete/rejected

/get

Publier un message vide dans cette rubrique pour obtenir le shadow d'appareil :

```
ShadowTopicPrefix/get
```

AWS IoT répond en publiant à l'un /get/accepted ou à l'autre/get/rejected.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get"
      ]
    }
  ]
}
```

/get/accepted

AWS IoT publie un document d'ombre de réponse dans cette rubrique lors du renvoi de l'ombre de l'appareil :

```
ShadowTopicPrefix/get/accepted
```

Pour de plus amples informations, veuillez consulter Documents d'état de la réponse.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/
accepted"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/
accepted"
      ]
    }
  ]
}
```

/get/rejected

AWS IoT publie un document de réponse aux erreurs dans cette rubrique lorsqu'il ne parvient pas à renvoyer l'ombre de l'appareil :

```
ShadowTopicPrefix/get/rejected
```

Pour de plus amples informations, veuillez consulter Document de réponse d'erreur.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/
rejected"
      ]
    },
    {
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/
rejected"
      ]
    }
  ]
}
```

/update

Publier un document d'état de la demande dans cette rubrique pour mettre à jour l'objet d'appareil :

```
ShadowTopicPrefix/update
```

Le corps du message contient un document d'état de demande partiel.

Un client tentant de mettre à jour l'état d'un appareil enverrait un document d'état de demande JSON avec la propriété `desired` comme la suivante :

```
{
  "state": {
    "desired": {
      "color": "red",
      "power": "on"
    }
  }
}
```

Un appareil mettant à jour son shadow enverrait un document d'état de demande JSON avec la propriété `reported`, comme ceci :

```
{
  "state": {
    "reported": {
      "color": "red",
      "power": "on"
    }
  }
}
```

AWS IoT répond en publiant à l'un /update/accepted ou à l'autre/update/rejected.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update"
      ]
    }
```

```
  ]
}
```

/update/delta

AWS IoT publie un document d'état de réponse dans cette rubrique lorsqu'il accepte une modification de l'ombre de l'appareil, et le document d'état de réponse contient des valeurs `desired` et des `reported` états différents :

```
ShadowTopicPrefix/update/delta
```

Le tampon de messages contient un /documents d'état de la réponse delta.

Détails du corps de message

- Un message publié dans `update/delta` comprend uniquement les attributs « souhaité » qui diffèrent entre les sections `desired` et `reported`. Il contient tous ces attributs, indépendamment qu'ils aient été contenus dans le message de mise à jour actuel ou qu'ils aient déjà été stockés dans AWS IoT. Les attributs qui ne diffèrent pas entre les sections `desired` et `reported` ne sont pas inclus.
- Si un attribut figure dans la section `reported`, mais qu'il n'a aucun équivalent dans la section `desired`, il n'est pas inclus.
- Si un attribut figure dans la section `desired`, mais qu'il n'a aucun équivalent dans la section `reported`, il n'est pas inclus.
- Si un attribut est supprimé de la section `reported`, mais qu'il existe toujours dans la section `desired`, il est inclus.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/delta"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/
delta"
      ]
    }
  ]
}
```

/update/accepted

AWS IoT publie un document d'état de réponse dans cette rubrique lorsqu'il accepte une modification de l'ombre de l'appareil :

```
ShadowTopicPrefix/update/accepted
```

Le tampon de messages contient un /document d'état de la réponse accepté.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/accepted"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/
accepted"
      ]
    }
  ]
}
```

/update/documents

AWS IoT publie un document d'état sur cette rubrique chaque fois qu'une mise à jour du shadow est effectuée avec succès :

```
ShadowTopicPrefix/update/documents
```

Le corps du message contient un /documents d'état de la réponse documents.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/documents"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/
documents"
      ]
    }
  ]
}
```

/update/rejected

AWS IoT publie un document de réponse aux erreurs dans cette rubrique lorsqu'il rejette une modification concernant l'ombre de l'appareil :

```
ShadowTopicPrefix/update/rejected
```

Le corps du message contient un Document de réponse d'erreur.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

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

```
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
update/rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/
rejected"
      ]
    }
  ]
}
```

/delete

Pour supprimer un shadow d'appareil, publiez un message vide dans la rubrique delete :

```
ShadowTopicPrefix/delete
```

Le contenu du message est ignoré.

Notez que la suppression d'une ombre ne rétablit pas son numéro de version à 0.

AWS IoT répond en publiant à l'un /delete/accepted ou à l'autre/delete/rejected.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete"
      ]
    }
  ]
}
```

/delete/accepted

AWS IoT publie un message dans cette rubrique lorsque l'ombre d'un appareil est supprimée :

```
ShadowTopicPrefix/delete/accepted
```

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
delete/accepted"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/
accepted"
      ]
    }
  ]
}
```

/delete/rejected

AWS IoT publie un document de réponse aux erreurs dans cette rubrique lorsqu'il ne parvient pas à supprimer l'ombre de l'appareil :

```
ShadowTopicPrefix/delete/rejected
```

Le corps du message contient un Document de réponse d'erreur.

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/
delete/rejected"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/
rejected"
      ]
    }
  ]
}
```

Documents du service Device Shadow

Le service Device Shadow respecte toutes les règles de la spécification JSON. Valeurs, objets et tableaux sont stockés dans le document shadow de l'appareil.

Table des matières

- Exemples de documents shadow
- Propriétés du document
- État Delta
- Documents shadow de gestion des versions
- Jetons clients dans les documents shadow
- Propriétés de document shadow vides
- Valeurs de tableau dans les documents shadow

Exemples de documents shadow

Le service Device Shadow utilise ces documents dans les opérations UPDATE, GET et DELETE à l'aide de l'API REST ou Pub/Sub des messages MQTT.

Exemples

- Document d'état de demande
- Documents d'état de la réponse
- Document de réponse d'erreur
- Document de réponse de liste de noms de shadows

Document d'état de demande

Un document d'état de demande a le format suivant :

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "clientToken": "token",
    "version": version
}
```

- `state` — Les mises à jour affectent uniquement les champs spécifiés. Généralement, vous utiliserez la propriété `reported` ou la propriété `desired`, mais pas les deux dans la même demande.
  - `desired` — Les propriétés d'état et les valeurs dont la mise à jour est demandé dans l'appareil.
  - `reported` — Les propriétés d'état et les valeurs signalées par l'appareil.
- `clientToken` — Si utilisé, vous pouvez faire correspondre la requête et la réponse correspondante par le jeton client.
- `version` — Le service Device Shadow traite la mise à jour uniquement si la version spécifiée correspond à la dernière version qu'il a.

Documents d'état de la réponse

Les documents d'état de la réponse ont le format suivant, selon le type de réponse.

/document d'état de la réponse accepté

```
{
```

```
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

/documents d'état de la réponse delta

```
{
    "state": {
        "attribute1": integer2,
        "attribute2": "string2",
        ...
        "attributeN": boolean2
    },
    "metadata": {
        "attribute1": {
            "timestamp": timestamp
        },
        "attribute2": {
            "timestamp": timestamp
        },
```

```
        ...
        "attributeN": {
            "timestamp": timestamp
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

/documents d'état de la réponse documents

```
{
  "previous" : {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
```

```
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version-1
},
"current": {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
```

```
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version
  },
  "timestamp": timestamp,
  "clientToken": "token"
}
```

Propriétés du document d'état de réponse

- `state` — Après une mise à jour réussie, il contient l’ de l'objet avant la mise à jour.
- `state` — Après une mise à jour réussie, il contient l’ de l'objet après la mise à jour.
- `state`
  - `reported` — Présent uniquement si un objet a déclaré des données dans la section et qu'il contient uniquement les champs qui se trouvaient dans le document d’état de la demande.
  - `desired` — Présente uniquement si un dispositif a communiqué des données dans la `desired` section et contient uniquement des champs qui figuraient dans le document d'état de la demande.
  - `delta` `desired` Présent uniquement si les données diffèrent des données actuelles du shadow.
- `metadata` — Contient les horodatages pour chaque attribut dans le `desired` et `reported` sections et afin que vous puissiez déterminer quand l'état a été mis à jour.
- `timestamp`— La date et l'heure d'époque auxquelles la réponse a été générée AWS IoT.
- `clientToken` — Présent uniquement si un jeton client a été utilisé lors de la publication d'un JSON valide dans la rubrique .
- `version` — Version actuelle du document du shadow de l'appareil partagé dans AWS IoT. Elle est augmentée d'une unité par rapport à la version précédente du document.

Document de réponse d'erreur

— Un document de réponse d'erreur a le format suivant :

```
{
    "code": error-code,
    "message": "error-message",
    "timestamp": timestamp,
    "clientToken": "token"
}
```

- `code` — Code de réponse HTTP qui indique le type d'erreur.
- `message` — Message texte qui fournit des informations supplémentaires.
- `timestamp`— Date et heure auxquelles la réponse a été générée AWS IoT. Cette propriété n'est pas présente dans tous les documents de réponse d'erreur.
- `clientToken`— Présent uniquement si un jeton client a été utilisé dans le message publié.

Pour de plus amples informations, veuillez consulter Messages d'erreur de Device Shadow.

Document de réponse de liste de noms de shadows

Un document de réponse de liste de noms de shadows a le format suivant :

```
{
    "results": [
        "shadowName-1",
        "shadowName-2",
        "shadowName-3",
        "shadowName-n"
    ],
    "nextToken": "nextToken",
    "timestamp": timestamp
}
```

- `results`— Le tableau des noms des ombres.
- `nextToken`— La valeur du jeton à utiliser dans les demandes paginées pour obtenir la page suivante de la séquence. Cette propriété n'est pas présente quand il ne reste plus de noms de shadows à renvoyer.

- `timestamp`— Date et heure auxquelles la réponse a été générée AWS IoT.

Propriétés du document

Un document de shadow d'appareil possède les propriétés suivantes :

`state`

`desired`

État souhaité de l'appareil. Les applications peuvent écrire dans cette partie du document pour mettre à jour l'état d'un appareil directement sans avoir à s'y connecter.

`reported`

État rapporté de l'appareil. Les appareils écrivent dans cette partie du document pour rapporter leur nouvel état. Les applications lisent cette partie du document pour déterminer le dernier état rapporté de l'appareil.

`metadata`

Informations sur les données stockées dans la section `state` du document. Il s'agit notamment des horodatages, en heure Unix, de chaque attribut de la section `state`, qui vous permet de déterminer quand ils ont été mis à jour.

> **Note**
>
> Les métadonnées ne contribuent pas à la taille du document pour les limites de service ou la tarification. Pour plus d'informations, consultez AWS IoT Limites de service .

`timestamp`

Indique quand le message a été envoyé par AWS IoT. En utilisant l'horodatage dans le message et les horodatages pour les attributs individuels dans la section `desired` ou `reported`, un appareil peut déterminer l'âge d'une propriété, même si l'appareil n'a pas d'horloge interne.

`clientToken`

Chaîne unique du dispositif qui permet d'associer les réponses à des demandes dans un environnement MQTT.

`version`

Version du document. Chaque fois que le document est mis à jour, ce numéro de version est incrémenté. Permet de s'assurer que la version du document en cours de mise à jour est la plus récente.

Pour de plus amples informations, veuillez consulter Exemples de documents shadow.

État Delta

L'état Delta est un type virtuel d'état qui contient l'écart entre les états `desired` et `reported`. Les champs de la section `desired` qui ne figurent pas dans la section `reported` sont inclus dans le delta. Les champs qui figurent dans la section `reported` mais pas dans la section `desired` ne sont pas inclus dans le delta. Le delta contient des métadonnées et ses valeurs sont égales aux métadonnées contenues dans le champ `desired`. Par exemple :

```
{
  "state": {
    "desired": {
      "color": "RED",
      "state": "STOP"
    },
    "reported": {
      "color": "GREEN",
      "engine": "ON"
    },
    "delta": {
      "color": "RED",
      "state": "STOP"
    }
  },
  "metadata": {
    "desired": {
      "color": {
        "timestamp": 12345
      },
      "state": {
        "timestamp": 12345
      }
      },
      "reported": {
```

```
        "color": {
          "timestamp": 12345
        },
        "engine": {
          "timestamp": 12345
        }
      },
      "delta": {
        "color": {
          "timestamp": 12345
        },
        "state": {
          "timestamp": 12345
        }
      }
    },
    "version": 17,
    "timestamp": 123456789
  }
}
```

Lorsque des objets imbriqués diffèrent, le delta contient le chemin d'accès à la racine.

```
{
  "state": {
    "desired": {
      "lights": {
        "color": {
          "r": 255,
          "g": 255,
          "b": 255
        }
      }
    },
    "reported": {
      "lights": {
        "color": {
          "r": 255,
          "g": 0,
          "b": 255
        }
      }
    },
```

```
    "delta": {
      "lights": {
        "color": {
          "g": 255
        }
      }
    }
  },
  "version": 18,
  "timestamp": 123456789
}
```

Le service Device Shadow calcule le delta en itérant sur chaque champ de l'état `desired` et en le comparant à l'état `reported`.

Les tableaux sont traités comme des valeurs. Si un tableau de la section `desired` ne correspond pas au tableau de la section `reported`, l'intégralité du tableau souhaité est copiée dans le delta.

Documents shadow de gestion des versions

Le service Device Shadow prend en charge la gestion des versions sur chaque message de mise à jour, qu'il s'agisse de la demande ou de la réponse. Cela signifie qu'à chaque mise à jour d'un shadow, la version du document JSON est incrémentée. Cela permet de garantir deux choses :

- Un client peut recevoir une erreur s'il tente de remplacer un shadow par un numéro de version plus ancien. Le client est informé qu'il doit effectuer une resynchronisation avant de mettre à jour un shadow d'appareil.
- Un client peut décider de ne pas agir sur un message reçu si le message est d'une version inférieure à la version stockée par le client.

Un client peut contourner la mise en correspondance des versions en n'incluant pas de version dans le document shadow.

Jetons clients dans les documents shadow

Vous pouvez utiliser un jeton client avec la messagerie MQTT pour vérifier si une demande et une réponse contiennent le même jeton client. Cela garantit que la réponse et la demande sont associées.

> **Note**
>
> La longueur du jeton client ne peut pas dépasser 64 octets. Un jeton client d'une longueur supérieure à 64 octets génère une réponse 400 (Demande erronée) et un message d'erreur ClientToken non valide.

Propriétés de document shadow vides

Les propriétés `reported` et `desired` d'un document shadow peuvent être vides ou omises lorsqu'elles ne s'appliquent pas à l'état actuel du shadow. Par exemple, un document shadow contient une propriété `desired` uniquement s'il a un état souhaité. Voici un exemple valide d'un document d'état sans propriété `desired` :

```
{
    "reported" : { "temp": 55 }
}
```

La propriété `reported` peut également être vide, par exemple si le shadow n'a pas été mis à jour par l'appareil :

```
{
    "desired" : { "color" : "RED" }
}
```

Si une mise à jour entraîne l'affectation de la valeur null aux propriétés `desired` ou `reported`, elle est supprimée du document. Voici comment supprimer la propriété `desired` en la définissant sur `null`. Vous pouvez le faire lorsqu'un appareil met à jour son état, par exemple.

```
{
    "state": {
        "reported": {
            "color": "red"
        },
        "desired": null
    }
}
```

Un document shadow peut également n'avoir aucune des propriétés `desired` et `reported`, auquel cas le document shadow est vide. Ceci est un exemple de document shadow vide, mais valide.

```
{
}
```

Valeurs de tableau dans les documents shadow

Les shadows prennent en charge les tableaux, mais les traitent comme des valeurs normales, dans la mesure où une mise à jour d'un tableau remplace l'intégralité du tableau. Il n'est pas possible de mettre à jour une partie d'un tableau.

État initial :

```
{
    "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
}
```

Mise à jour:

```
{
    "desired" : { "colors" : ["RED"] }
}
```

État final :

```
{
    "desired" : { "colors" : ["RED"] }
}
```

Les tableaux ne peuvent pas avoir de valeurs null. Par exemple, le tableau suivant n'est pas valide et sera rejeté.

```
{
    "desired" : {
        "colors" : [ null, "RED", "GREEN" ]
    }
}
```

Messages d'erreur de Device Shadow

Le service Device Shadow publie un message dans la rubrique d'erreur (via MQTT) en cas d'échec d'une tentative de modifier le document d'état. Ce message est émis uniquement en réponse à une demande de publication dans l'une des `$aws` rubriques réservées. Si le client met à jour le document à l'aide de l'API REST, il reçoit le code d'erreur HTTP dans le cadre de la réponse, et aucun message d'erreur MQTT n'est émis.

Code d’erreur HTTP	Messages d’erreur
400 (Requête erronée)	• JSON non valide • Nœud requis manquant : état • Le nœud d'état doit être un objet • Le nœud souhaitée doit être un objet • Le nœuds déclaré doit être un objet • Version non valide • ClientToken non valide **Note** Un jeton client d'une longueur supérieure à 64 octets génère cette réponse. • Le code JSON contient trop de niveaux d'imbrication ; le maximum est 6 • L'état contient un nœud non valide
401 (Accès non autorisé)	• Non autorisé
403 (Accès interdit)	• Accès interdit
404 (Introuvable)	• Objet non trouvé • Il n'existe aucune ombre portant le nom : *shadowName*
409 (Conflit)	• Conflit de versions

Code d’erreur HTTP	Messages d’erreur
413 (Charge utile trop importante)	• La charge utile dépasse la taille maximale autorisée
415 (Type de support non pris en charge)	• Codage documenté non pris en charge ; l'encodage pris en charge est UTF-8
429 (Nombre de requêtes trop élevé)	• Le service Device Shadow génère ce message d'erreur lorsqu'il y a plus de 10 demandes en vol sur une seule connexion. Une demande en vol est une demande en cours qui a été lancée mais qui n'est pas encore terminée.
500 (Erreur interne du serveur)	• Échec du service interne

AWS IoT Device Management Catalogue de packages logiciels

Avec AWS IoT Device Management Software Package Catalog, vous pouvez conserver un inventaire des packages logiciels et de leurs versions. Vous pouvez associer des versions de package à des objets individuels et à des groupes d'objets AWS IoT dynamiques, et les déployer par le biais de processus ou de AWS IoT tâches internes.

Un package logiciel contient une ou plusieurs versions de package, qui constituent un ensemble de fichiers qui peuvent être déployés en tant qu'une unité unique. Les versions du package peuvent contenir le microprogramme, les mises à jour du système d'exploitation, les applications de l'appareil, les configurations et les correctifs de sécurité. Au fur et à mesure que le logiciel évolue, vous pouvez créer une nouvelle version du package et la déployer dans votre flotte.

Le hub de AWS IoT progiciels se trouve à l'intérieur AWS IoT Core. Vous pouvez utiliser le hub pour enregistrer et gérer de manière centralisée l'inventaire et les métadonnées de vos progiciels, ce qui crée un catalogue des packages logiciels et de leurs versions. Vous pouvez choisir de regrouper les appareils en fonction des packages logiciels et des versions de packages déployés sur l'appareil. Cette fonctionnalité permet de conserver l'inventaire des packages côté appareil sous la forme d'une ombre nommée, d'associer et de regrouper les appareils en fonction des versions, et de visualiser la distribution des versions de packages au sein du flotte à l'aide des métriques de la flotte.

Si vous disposez d'un système interne de déploiement de logiciels, vous pouvez continuer à utiliser ce processus pour déployer les versions de vos packages. Si vous n'avez pas établi de processus de déploiement ou si vous préférez, nous vous recommandons d'utiliser des AWS IoT tâches pour utiliser les fonctionnalités du catalogue des packages logiciels. Pour plus d'informations, consultez la section Préparation AWS IoT des tâches.

Ce chapitre contient les sections suivantes :

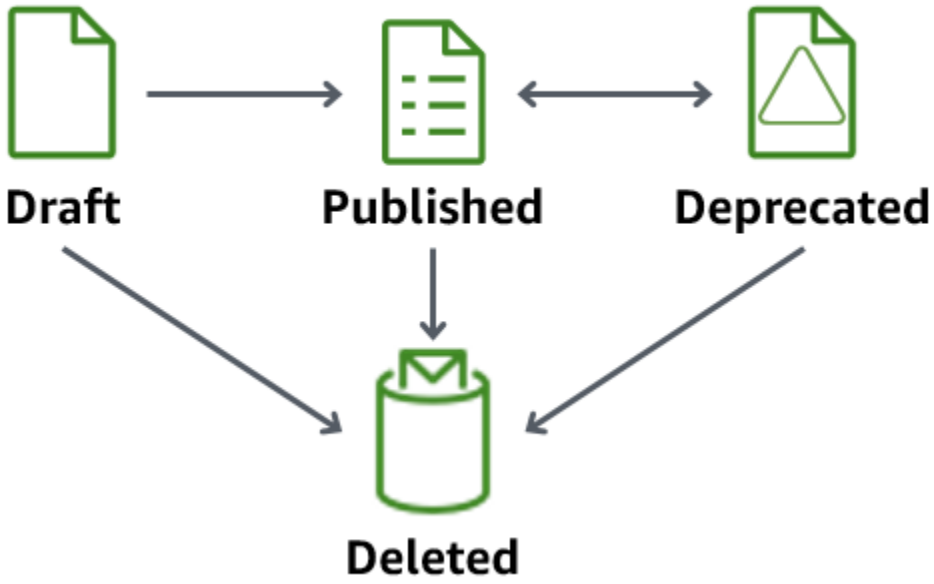
- Préparation à l'utilisation du Catalogue de Logiciels
- Préparation de la sécurité
- Préparation de l'indexation de la flotte
- Préparation des AWS IoT emplois
- Démarrage avec le Catalogue de Logiciels

Préparation à l'utilisation du Catalogue de Logiciels

La section suivante fournit une vue d'ensemble du cycle de vie des versions du package et des informations sur l'utilisation du catalogue de packages AWS IoT Device Management logiciels.

Cycle de vie des versions du package

Une version d'un package peut évoluer selon les états du cycle de vie suivants : `draft`, `published`, et `deprecated`. Cela peut aussi l'être `deleted`.



- Ébauche

  Lorsque vous créez une version de package, elle est dans un `draft` état. Cet état indique que le package logiciel est en cours de préparation ou qu'il est incomplet.

  Tant que la version du package est dans cet état, vous ne pouvez pas la déployer. Vous pouvez modifier la description, les attributs et les balises de la version du package.

  Vous pouvez effectuer la transition d'une version de package `draft` existante `published` ou existante en `deleted` utilisant la console, ou en exécutant des opérations d'DeletePackageVersionAPI UpdatePackageVersionou d'API.

- Publié

  Lorsque la version de votre package est prête à être déployée, passez de la version du package à un `published` état. Dans cet état, vous pouvez choisir d'identifier la version du package

comme version par défaut en modifiant le package logiciel dans la console ou via l'opération UpdatePackageAPI. Dans cet état, vous ne pouvez modifier que la description et les balises.

Vous pouvez effectuer la transition d'une version de package `published` existante `deprecated` ou existante en `deleted` utilisant la console ou en exécutant des opérations d'DeletePackageVersionAPI UpdatePackageVersionou d'API.

- Obsolète

  Si une nouvelle version de package est disponible, vous pouvez transférer les versions antérieures de package vers `deprecated`. Vous pouvez toujours déployer des tâches avec une version de package obsolète. Vous pouvez également nommer une version de package obsolète comme version par défaut et modifier uniquement la description et les balises.

  Envisagez de transférer la version d'un package vers une version obsolète, mais que vous avez toujours des appareils sur le terrain qui utilisent l'ancienne version ou que vous devez la maintenir en raison d'une dépendance au `deprecated` moment de l'exécution.

  Vous pouvez effectuer la transition d'une version de package `deprecated` existante `published` ou existante en `deleted` utilisant la console ou en exécutant des opérations d'DeletePackageVersionAPI UpdatePackageVersionou d'API.

- Supprimé

  Lorsque vous n'avez plus l'intention d'utiliser une version de package, vous pouvez la supprimer à l'aide de la console ou en lançant l'opération DeletePackageVersionAPI.

  > **Note**
  >
  > Si vous supprimez une version de package alors que des tâches en attente y font référence, vous recevrez un message d'erreur lorsque la tâche sera terminée avec succès et que vous tenterez de mettre à jour l'ombre réservée nommée.
  > Si la version du package logiciel que vous souhaitez supprimer est nommée version du package par défaut, vous devez d'abord mettre à jour le package pour nommer une autre version par défaut ou laisser le champ anonyme. Vous pouvez le faire à l'aide de la console ou de l'opération UpdatePackageVersionAPI. (Pour supprimer une version de package nommée par défaut, définissez le unsetDefaultVersionparamètre sur true lorsque vous lancez l'opération UpdatePackaged'API).

Si vous supprimez un package logiciel via la console, toutes les versions du package associées à ce package sont supprimées, sauf si l'une d'entre elles est désignée comme version par défaut.

Conventions de dénomination des versions du package

Lorsque vous nommez des versions de package, il est important de planifier et d'appliquer une stratégie de dénomination logique afin que vous et les autres puissiez facilement identifier la dernière version du package et la progression des versions. Vous devez fournir un nom de version lors de la création de la version du package, mais la stratégie et le format dépendent en grande partie de votre analyse de rentabilisation.

À titre de bonne pratique, nous vous recommandons d'utiliser le format de versionnement SemVersémantique. Par exemple, `1.2.3` où `1` se trouve la version majeure pour les modifications fonctionnellement incompatibles, `2` la version majeure pour les modifications fonctionnellement compatibles et `3` la version corrective (pour les corrections de bogues). Pour plus d'informations, veuillez consulter la rubrique Gestion des versions sémantique 2.0.0. Pour plus d'informations sur les exigences relatives au nom de version du package, consultez VersionName dans le guide de référence de l' AWS IoT API.

Version par défaut

La définition d'une version par défaut est facultative. Vous pouvez ajouter ou supprimer des versions de package par défaut. Vous pouvez également déployer une version de package qui n'est pas nommée version par défaut.

Lorsque vous créez une version de package, elle est placée dans un `draft` état et ne peut pas être nommée version par défaut tant que vous n'avez pas fait passer la version du package à la version publiée. Le Catalogue de Logiciels ne sélectionne pas automatiquement une version par défaut ni ne met à jour une version de package plus récente par défaut. Vous devez nommer intentionnellement la version du package que vous choisissez via la console ou en lançant l'opération UpdatePackageVersiond'API.

Attributs de version

Les attributs de version et leurs valeurs contiennent des informations importantes sur les versions de vos packages. Nous vous recommandons de définir des attributs généraux pour un package ou

une version de package. Par exemple, vous pouvez créer une paire nom-valeur pour la plate-forme, l'architecture, le système d'exploitation, la date de sortie, l'auteur ou l'URL Amazon S3.

Lorsque vous créez une AWS IoT tâche avec un document de tâche, vous pouvez également choisir d'utiliser une variable de substitution (`$parameter`) qui fait référence à la valeur d'un attribut. Pour plus d'informations, consultez la section Préparation AWS IoT des tâches.

Les attributs de version utilisés dans les versions de package ne seront pas automatiquement ajoutés à l'ombre nommée réservée et ne peuvent pas être indexés ou interrogés directement via Fleet Indexing. Pour indexer ou interroger les attributs de version d'un package via Fleet Indexing, vous pouvez renseigner l'attribut de version dans l'ombre nommée réservée.

Nous recommandons que le paramètre d'attribut de version figurant dans le périphérique de capture d'ombres réservé indique les propriétés, telles que le système d'exploitation et l'heure d'installation. Ils peuvent également être indexés et interrogés via Fleet Indexing.

Les attributs de version ne sont pas tenus de respecter une convention de dénomination spécifique. Vous pouvez créer des paires nom-valeur pour répondre aux besoins de votre entreprise. La taille combinée de tous les attributs d'une version de package est limitée à 3KB. Pour plus d'informations, veillez consultez Software Package Catalog software package and package versions limits.

Utilisation de tous les attributs dans un document de travail

Vous pouvez faire en sorte que tous les attributs de version du package soient automatiquement ajoutés à votre déploiement de tâches pour les appareils sélectionnés. Pour utiliser automatiquement tous les attributs de version du package par programmation dans une API ou une commande CLI, reportez-vous à l'exemple de document de tâche suivant :

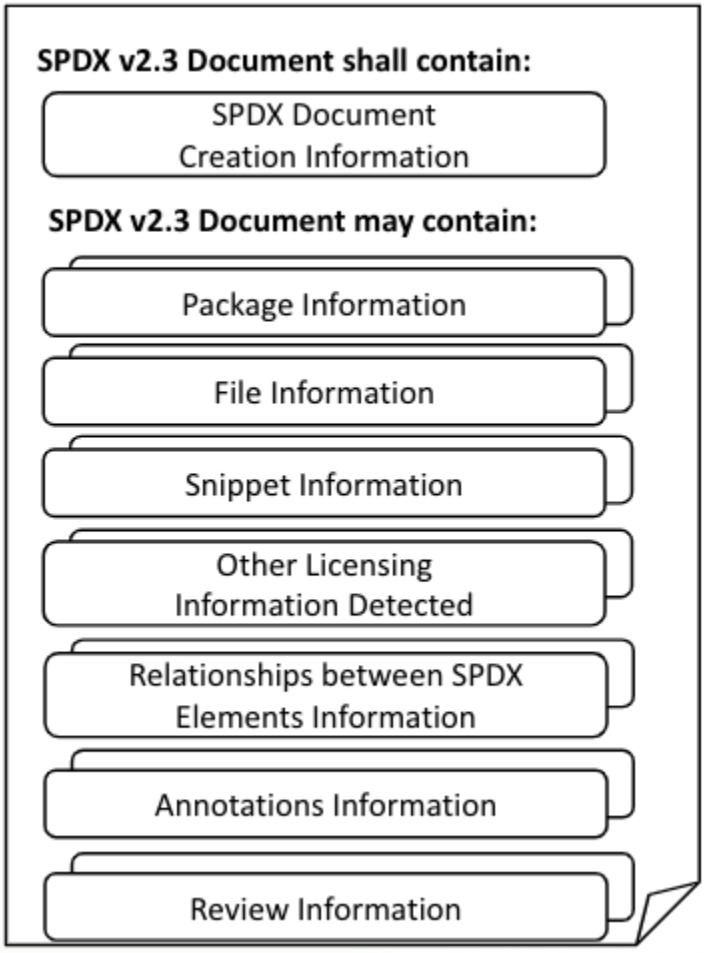
```
"TestPackage": "${aws:iot:package:TestPackage:version:PackageVersion:attributes}"
```

Nomenclature du logiciel

La nomenclature logicielle (SBOM) fournit un référentiel central pour tous les aspects de votre progiciel. Outre le stockage des packages logiciels et des versions des packages, vous pouvez stocker la nomenclature logicielle (SBOM) associée à chaque version de AWS IoT Device Management package dans le catalogue des packages logiciels. Un package logiciel contient une ou plusieurs versions de package et chaque version de package comprend un ou plusieurs composants. Chacun de ces composants prenant en charge la composition d'une version de package spécifique peut être décrit et catalogué à l'aide d'une nomenclature logicielle. Les normes industrielles relatives aux nomenclatures logicielles prises en charge sont SPDX et CycloneDX. Lorsqu'un SBOM est créé

pour la première fois, il est validé par rapport aux formats standard du secteur SPDX et CycloneDX. Pour plus d'informations sur le SPDX, consultez System Package Data Exchange. Pour plus d'informations sur CycloneDx, voir CycloneDx.

La nomenclature du logiciel décrit tous les aspects des composants d'une version de package spécifique, tels que les informations sur le package, les informations sur les fichiers et les autres métadonnées pertinentes. Consultez l'exemple ci-dessous de structure de document de nomenclature logicielle au format SPDX :



Avantages de la nomenclature logicielle

L'un des principaux avantages de l'ajout de votre nomenclature logicielle pour une version de package dans le catalogue de packages logiciels est la gestion des vulnérabilités.

Gestion des vulnérabilités

L'évaluation et l'atténuation de votre vulnérabilité aux risques de sécurité apparents liés aux composants logiciels restent essentiels pour protéger l'intégrité de votre parc d'appareils. En ajoutant

la nomenclature logicielle stockée dans le catalogue des packages logiciels pour chaque version de package, vous pouvez identifier de manière proactive les failles de sécurité en identifiant les appareils à risque en fonction de la version de leur package et du SBOM à l'aide de votre propre solution de gestion des vulnérabilités interne. Vous pouvez déployer des correctifs sur les appareils concernés et protéger votre parc d'appareils.

Stockage de la nomenclature logicielle

La nomenclature logicielle (SBOM) de chaque version du package logiciel est stockée dans un compartiment Amazon S3 à l'aide de la fonctionnalité de gestion des versions d'Amazon S3. Le compartiment Amazon S3 stockant le SBOM doit être situé dans la même région que celle où la version du package a été créée. Un compartiment Amazon S3 utilisant la fonctionnalité de gestion des versions conserve plusieurs variantes d'un objet dans le même compartiment. Pour plus d'informations sur l'utilisation de la gestion des versions dans un compartiment Amazon S3, consultez Utilisation de la gestion des versions dans les compartiments Amazon S3.

> **Note**
>
> Chaque version du progiciel peut être associée à plusieurs fichiers SBOM, mais les fichiers SBOM doivent être stockés dans un seul fichier d'archive zip.

La clé Amazon S3 et l'ID de version spécifiques à votre compartiment sont utilisés pour identifier de manière unique chaque version d'une nomenclature logicielle pour une version de package.

> **Note**
>
> Pour une version de package contenant un seul fichier SBOM, vous pouvez stocker ce fichier SBOM dans votre compartiment Amazon S3 sous forme de fichier d'archive zip.
> Pour une version de package contenant plusieurs fichiers SBOM, vous devez placer tous les fichiers SBOM dans un seul fichier d'archive zip, puis stocker ce fichier d'archive zip dans votre compartiment Amazon S3.
> Tous les fichiers SBOM stockés dans le fichier d'archive zip unique dans les deux scénarios sont formatés sous forme de fichiers SPDX ou CyclonedX .json.

Politique d'autorisations

Pour AWS IoT agir en tant que principal désigné pour accéder aux fichiers d'archive zip SBOM stockés dans le compartiment Amazon S3, vous avez besoin d'une politique d'autorisation basée sur les ressources. Reportez-vous à l'exemple suivant pour connaître la bonne politique d'autorisation basée sur les ressources :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "iot.amazonaws.com"
                ]
            },
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::bucketName/*"
        }
    ]
}
```

Pour plus d'informations sur les politiques d'autorisation basées sur les ressources, voir AWS IoT Politiques basées sur les ressources

Mettre à jour le SBOM

Vous pouvez mettre à jour la nomenclature logicielle aussi souvent que nécessaire pour protéger et améliorer votre parc d'appareils. Chaque fois que la nomenclature du logiciel est mise à jour dans votre compartiment Amazon S3, l'ID de version change et vous devez associer la nouvelle URL du compartiment Amazon S3 à la version du package logiciel appropriée. Vous verrez le nouvel ID de version dans la colonne Amazon S3 Object version ID sur la page de version du package dans le AWS Management Console. En outre, vous pouvez utiliser l'opération API `GetPackageVersion` ou la commande CLI `get-package-version` pour afficher le nouvel ID de version.

> **Note**
>
> La mise à jour de votre nomenclature logicielle, qui entraînera l'attribution d'un nouvel identifiant de version, n'entraînera pas la création d'une nouvelle version du package.

Pour plus d'informations sur les clés d'objet Amazon S3, consultez Création de noms de clés d'objets.

Activation de l'indexation AWS IoT de la flotte

L'activation AWS IoT de l'indexation du parc est obligatoire pour utiliser le catalogue de packages AWS IoT Device Management logiciels. Pour tirer parti de l'indexation du AWS IoT parc à l'aide du Software Package Catalog, définissez le nom réservé shadow (`$package`) comme source de données pour chaque appareil sur lequel vous souhaitez indexer et recueillir des métriques. Pour plus d'informations sur les ombres nommées réservées, consultezOmbre nommée réservée.

L'indexation du parc fournit un support qui permet de regrouper AWS IoT les objets par le biais de groupes d'objets dynamiques filtrés par version de progiciel. Par exemple, l'indexation de la flotte peut identifier les éléments pour lesquels une version de package spécifique est installée ou non, pour laquelle aucune version de package n'est installée ou qui correspondent à des paires nom-valeur spécifiques. Enfin, l'indexation du parc fournit des indicateurs standard et personnalisés que vous pouvez utiliser pour avoir un aperçu de l'état de votre parc d'appareils. Pour de plus amples informations, veuillez consulter Préparation de l'indexation de la flotte.

> **Note**
>
> L'activation de l'indexation du parc pour le Catalogue de Logiciels entraîne des coûts de service standard. Pour plus d'informations, consultez AWS IoT Device Management Pricing

Ombre nommée réservée

Le Ombre nommée réservée`$package`, reflète l'état des packages logiciels et des versions de packages installés sur l'appareil. L'indexation de flotte utilise l'ombre nommée réservée comme source de données pour créer des métriques standard et personnalisées afin que vous puissiez interroger l'état de votre flotte. Pour de plus amples informations, veuillez consulter Preparing fleet indexing.

Une ombre nommée réservée est similaire à une ombre nommée, sauf que son nom est prédéfini et que vous ne pouvez pas le modifier. En outre, l'ombre nommée réservée n'est pas mise à jour avec les métadonnées et utilise uniquement les `attributes` et `version` mots clés.

Les demandes de mise à jour qui incluent d'autres mots clés, tels que `description`, recevront une réponse d'erreur dans le `rejected` sujet. Pour plus d'informations, consultez Device Shadow error messages.

Il peut être créé lorsque vous créez un AWS IoT objet via la console, lorsqu'une AWS IoT tâche se termine avec succès et met à jour le shadow, et si vous lancez l'opération d'`UpdateThingShadow`API. Pour plus d'informations, consultez UpdateThingShadowle guide du AWS IoT Core développeur.

> **Note**
>
> L'indexation de l'ombre nommée réservée n'est pas prise en compte dans le nombre d'ombres nommées que l'indexation de flotte peut indexer. Pour plus d'informations, consultez AWS IoT Device Management fleet indexing limits and quotas. En outre, si vous choisissez de faire en sorte que les AWS IoT jobs mettent à jour l'ombre nommée réservée lorsqu'une tâche est terminée avec succès, l'appel d'API est comptabilisé dans votre Device Shadow et vos opérations de registre et peut entraîner un coût. Pour plus d'informations, consultez les rubriques Limites et quotas des AWS IoT Device Management tâches ainsi que le type de données de l'IndexingFilterAPI.

Structure de l'**$package**ombre

L'ombre nommée réservée contient les éléments suivants :

```
{
    "state": {
        "reported": {
            "<packageName>": {
                "version": "",
                "attributes": {
                }
            }
        }
    },
    "version" : 1
```

```
    "timestamp" : 1672531201
}
```

Les propriétés de l'ombre sont mises à jour avec les informations suivantes :

- `<packageName>` : nom du package logiciel installé, qui est mis à jour avec le paramètre PackageName.
- `version` : nom de la version du package installé, qui est mise à jour avec le paramètre versionName.
- `attributes` : métadonnées facultatives stockées par l'appareil et indexées par l'indexation de la flotte. Cela permet aux clients d'interroger leurs index en fonction des données stockées.
- `version` : le numéro de version de l’ombre. Elle est automatiquement incrémentée chaque fois que l'ombre est mise à jour et commence à 1.
- `timestamp` :Indique quand l'ombre a été mise à jour pour la dernière fois et est enregistrée à l'heure Unix.

Pour plus d'informations sur le format et le comportement d'une ombre nommée, consultez Service AWS IoT Device Shadow Ordre des messages.

Suppression d'un package logiciel et de ses versions

Avant de supprimer un package logiciel, effectuez les opérations suivantes :

- Vérifiez que le package et ses versions ne sont pas activement déployés.
- Supprimez d'abord toutes les versions associées. Si l'une des versions est désignée comme version par défaut, vous devez supprimer la version par défaut nommée du package. La désignation d'une version par défaut étant facultative, il n'y a aucun conflit lors de sa suppression. Pour supprimer la version par défaut du package logiciel, modifiez le package via la console ou utilisez l'opération UpdatePackageVersionAPI.

Tant qu'il n'existe pas de version de package par défaut nommée, vous pouvez utiliser la console pour supprimer un package logiciel et toutes ses versions de package seront également supprimées. Si vous utilisez un appel d'API pour supprimer des packages logiciels, vous devez d'abord supprimer les versions des packages, puis le package logiciel.

Préparation de la sécurité

Cette section décrit les principales exigences de sécurité pour le catalogue de packages AWS IoT Device Management logiciels.

Authentification basée sur les ressources

Le Catalogue de Logiciels utilise une autorisation basée sur les ressources pour renforcer la sécurité lors de la mise à jour des logiciels de votre flotte. Cela signifie que vous devez créer une politique AWS Identity and Access Management (IAM) qui accorde les droits d'exécution`create`,, `read` `update``delete`, et d'`list`actions pour les packages logiciels et les versions de packages, et référencer les packages logiciels et les versions de package spécifiques que vous souhaitez déployer dans la `Resources` section. Vous avez également besoin de ces droits pour pouvoir mettre à jour l'ombre nommée réservée. Vous référencez les packages logiciels et les versions des packages en incluant un Amazon Resource Name (ARN) pour chaque entité.

> **Note**
>
> Si vous souhaitez que la politique accorde des droits pour les appels d'API de version de package (tels que CreatePackageVersionUpdatePackageVersion,, DeletePackageVersion), vous devez inclure à la fois le package logiciel et la version du package ARNs dans la politique. Si vous souhaitez que la politique accorde des droits pour les appels d'API des packages logiciels (tels que CreatePackageUpdatePackage, et DeletePackage), vous devez inclure uniquement l'ARN du package logiciel dans la politique.

Structurez le package logiciel et la version du package ARNs comme suit :

- Package logiciel : `arn:aws:iot:<region>:<accountID>:package/<packageName>/package`
- Version du package : `arn:aws:iot:<region>:<accountID>:package/<packageName>/version/<versionName>`

> **Note**
>
> Il existe d'autres droits connexes que vous pouvez inclure dans cette politique. Par exemple, vous pouvez inclure un ARN pour le `job`, `thinggroup`, et`jobtemplate`. Pour plus

d'informations et une liste complète des options de politique, consultez la section Sécurisation des utilisateurs et des appareils avec AWS IoT Jobs.

Par exemple, si vous disposez d'un package logiciel et d'une version de package nommés comme suit :

- AWS IoT chose : `myThing`
- Nom du package : `samplePackage`
- Version `1.0.0`

La politique doit ressembler à l'exemple suivant :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:createPackage",
                "iot:createPackageVersion",
                "iot:updatePackage",
                "iot:updatePackageVersion"
            ],
            "Resource": [
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage",
               "arn:aws:iot:us-east-1:111122223333:package/samplePackage/
version/1.0.0"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": "arn:aws:iot:us-east-1:111122223333:thing/myThing/
$package"
```

```
        }
    ]
}
```

AWS IoT Droits de travail pour déployer des versions de packages

Pour des raisons de sécurité, il est important que vous accordiez les droits de déploiement de packages et de versions de packages, et que vous nommiez les packages et versions de packages spécifiques qu'ils sont autorisés à déployer. Pour ce faire, vous créez un rôle et une politique IAM qui autorisent le déploiement de tâches avec des versions de package. La politique doit spécifier les versions du package de destination en tant que ressource.

Politique IAM

La politique IAM accorde le droit de créer une tâche qui inclut le package et la version nommés dans la `Resource` section.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJob",
                "iot:CreateJobTemplate"
            ],
            "Resource":[
                "arn:aws:iot:*:111122223333:job/<jobId>",
                "arn:aws:iot:*:111122223333:thing/<thingName>/$package",
                "arn:aws:iot:*:111122223333:thinggroup/<thingGroupName>",
                "arn:aws:iot:*:111122223333:jobtemplate/<jobTemplateName>",
                "arn:aws:iot:*:111122223333:package/<packageName>/
version/<versionName>"
            ]
        }
    ]
}
```

> **Note**
>
> Si vous souhaitez déployer une tâche qui désinstalle un package logiciel et une version de package, vous devez autoriser un ARN contenant la version du package`$null`, comme dans le cas suivant :

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

AWS IoT Droits de travail pour mettre à jour l'ombre nommée réservée

Pour permettre aux tâches de mettre à jour le nom réservé de l'objet lorsque la tâche est terminée avec succès, vous devez créer un rôle et une politique IAM. Il existe deux manières de procéder dans la AWS IoT console. La première consiste à créer un package logiciel dans la console. Si la boîte de dialogue Activer les dépendances pour la gestion des packages s'affiche, vous pouvez choisir d'utiliser un rôle existant ou d'en créer un nouveau. Dans la AWS IoT console, vous pouvez également choisir Gérer l'indexation dans Paramètres, puis Gérer l'indexation pour les packages et les versions des appareils.

> **Note**
>
> Si vous choisissez de demander au service AWS IoT Job de mettre à jour l'ombre nommée réservée lorsqu'une tâche est terminée avec succès, l'appel d'API est pris en compte dans vos opérations de Device Shadow et de registre et peut entraîner un coût. Pour en savoir plus, consultez Pricing AWS IoT Core (Tarification).

Lorsque vous utilisez l'option Créer un rôle, le nom du rôle généré commence par `aws-iot-role-update-shadows` et contient les politiques suivantes :

Configuration de rôles

Autorisations

La politique d'autorisation accorde le droit d'interroger et de mettre à jour l’ombre d’objet. Le `$package` paramètre de l'ARN de la ressource cible l'ombre nommée réservée.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iot:DescribeEndpoint",
            "Resource": ""
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Resource": [
                "arn:aws:iot:<regionCode>:111122223333:thing/<thingName>/
$package"
            ]
        }
    ]
}
```

Relation d'approbation

Outre la politique d'autorisations, le rôle nécessite une relation de confiance avec AWS IoT Core enfin que entité peut assumer le rôle et mettre à jour le nom d'ombre réservé.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Configuration d'une politique utilisateur

iam : autorisation PassRole

Enfin, vous devez être autorisé à transmettre le rôle AWS IoT Core lorsque vous appelez l'opération UpdatePackageConfigurationd'API.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iot:UpdatePackageConfiguration"
            ],
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
        }
    ]
}
```

AWS IoT Autorisations de téléchargement des jobs depuis Amazon S3

Le document de travail est enregistré dans Amazon S3. Vous vous référez à ce fichier lorsque vous expédiez via AWS IoT Jobs. Vous devez fournir à AWS IoT Jobs le droit de télécharger le fichier (`s3:GetObject`). Vous devez également établir une relation de confiance entre Amazon S3 et AWS IoT Jobs. Pour obtenir des instructions sur la création de ces politiques, voir Presigned URLs dans Gestion des tâches.

Autorisations de mise à jour de la nomenclature du logiciel pour une version de package

Pour mettre à jour la nomenclature logicielle d'une version de package dans l'état `DraftPublished`, ou dans l'état `Deprecated` du cycle de vie, vous avez besoin d'un AWS Identity and Access Management rôle et de politiques permettant de localiser la nouvelle nomenclature logicielle dans Amazon S3 et de mettre à jour la version du package dans AWS IoT Core.

Tout d'abord, vous allez placer la nomenclature logicielle mise à jour dans votre compartiment Amazon S3 versionné et appeler l'opération d'[UpdatePackageVersion]API avec le `sboms` paramètre inclus. Ensuite, votre principal autorisé assumera le rôle IAM que vous avez créé, localisera la nomenclature logicielle mise à jour dans Amazon S3 et mettra à jour la version du package dans le catalogue AWS IoT Core de packages logiciels.

Les règles suivantes sont requises pour effectuer cette mise à jour :

Stratégies

- Politique de confiance Politique établissant une relation de confiance avec le principal autorisé assumant le rôle IAM afin qu'il puisse localiser la nomenclature logicielle mise à jour depuis votre compartiment versionné dans Amazon S3 et mettre à jour la version du package dans. AWS IoT Core
  - JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

  - JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

- Politique d'autorisations : politique d'accès au compartiment versionné Amazon S3 dans lequel la nomenclature logicielle est stockée pour une version de package et de mise à jour de la version du package dans AWS IoT Core.
  - JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1"
            ]
        }
    ]
}
```

  - JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdatePackageVersion"
            ],
            "Resource": [
                "arn:aws:iot:*:111122223333:package/<packageName>/
version/<versionName>"
            ]
        }
    ]
}
```

- Transmettre les autorisations de rôle : politique octroyant l'autorisation de transmettre le rôle IAM à Amazon S3 et AWS IoT Core lorsque vous appelez l'opération `UpdatePackageVersion` d'API.
  - JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "s3:GetObject"
            ],
                "arn:aws:s3:::awsexamplebucket1"
        }
    ]
}
```

  - JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iot:UpdatePackageVersion"
            ],
            "Resource": "arn:aws:iam::111122223333:role/<roleName>"
        }
    ]
}
```

> **Note**
>
> Vous ne pouvez pas mettre à jour la nomenclature du logiciel sur une version du package qui est passée à l'état de `Deleted` cycle de vie.

Pour plus d'informations sur la création d'un rôle IAM pour un AWS service, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service.

Pour plus d'informations sur la création d'un compartiment Amazon S3 et le téléchargement d'objets dans celui-ci, consultez les sections Création d'un compartiment et Chargement d'objets.

Préparation de l'indexation de la flotte

Grâce à l'indexation des AWS IoT flottes, vous pouvez rechercher et agréger des données en utilisant le nom réservé shadow (`$package`). Vous pouvez également regrouper AWS IoT des objets en interrogeant les groupes Ombre nommée réservée d'objets dynamiques. Par exemple, vous pouvez trouver des informations sur AWS IoT les éléments qui utilisent une version de package spécifique, sur lesquels aucune version de package spécifique n'est installée ou sur lesquels aucune version de package n'est installée. Vous pouvez obtenir des informations supplémentaires en combinant les attributs. Par exemple, identifier les objets dotés d'une version spécifique et d'un type d'objet spécifique (tels que la version 1.0.0 et le type d'objet pump_sensor). Pour plus d'informations, veuillez consulter la rubrique Fleet indexing.

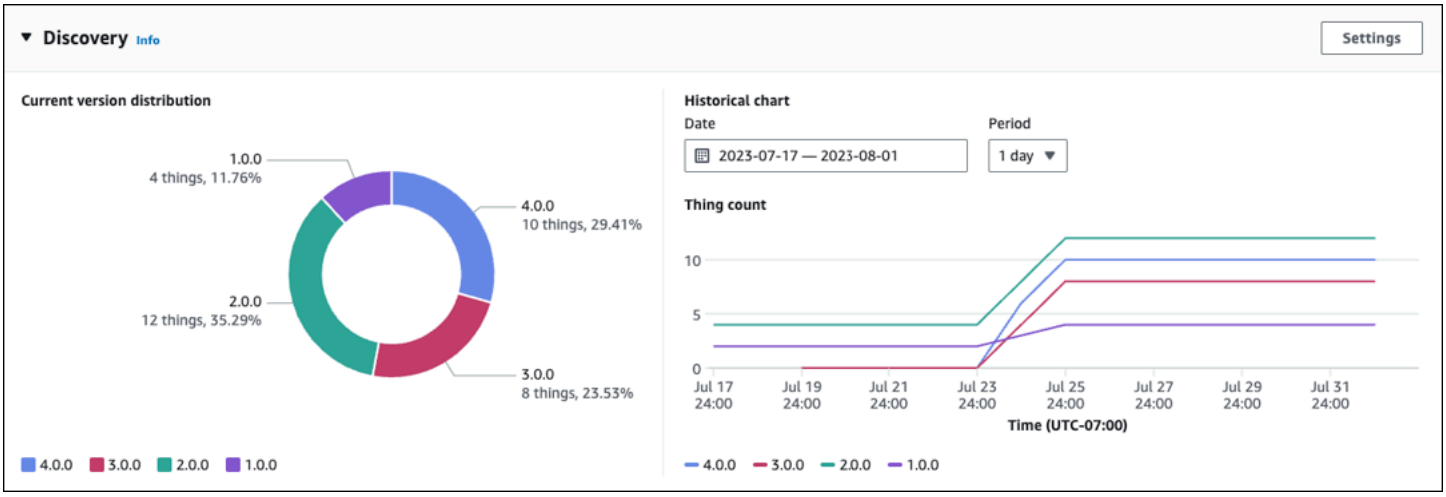
Définir l'**`$package`**ombre comme source de données

Pour utiliser l'indexation de flotte avec le Catalogue de Logiciels, vous devez activer l'indexation de flotte, définir l'ombre nommée comme source de données et définir `$package` comme filtre d'ombre nommé. Si vous n'avez pas activé l'indexation de la flotte, vous pouvez l'activer dans le cadre de ce processus. A partir deAWS IoT Core dans la console, ouvrez Paramètres, choisissez Gérer l'indexation, puis Ajouter des ombres nommées, Ajouter des packages logiciels et des versions de l'appareil, et Mettre à jour. Pour de plus amples informations, veuillez consulter la section Gestion de l'indexation des objets.

Vous pouvez également activer l'indexation de la flotte lorsque vous créez votre premier package. Lorsque la boîte de dialogue Activer les dépendances pour la gestion des packages apparaît, choisissez l'option permettant d'ajouter les packages logiciels et les versions des appareils en tant que sources de données pour l'indexation de la flotte. En sélectionnant cette option, vous activez également l'indexation de la flotte.

> **Note**
>
> L'activation de l'indexation de la flotte pour le Catalogue de Logiciels entraîne des coûts de service standard. Pour plus d'informations, consultez AWS IoT Device Management Pricing

Métriques affichées dans la console



Sur la page de détails du package logiciel de la AWS IoT console, le panneau Discovery affiche les métriques standard ingérées dans l'`$package`ombre.

- Le tableau de distribution des versions actuelles indique le nombre d'appareils et le pourcentage des 10 versions de package les plus récentes associées à un AWS IoT objet parmi tous les appareils associés à ce package logiciel. Remarque : Si le package logiciel comporte plus de versions que celles indiquées dans le tableau, vous pouvez les trouver regroupées dans la section Autres.
- Le graphique historique indique le nombre d'appareils associés aux versions de package sélectionnées sur une période donnée. Le graphique est initialement vide jusqu'à ce que vous sélectionniez jusqu'à 5 versions de package et que vous définissiez la plage de dates et l'intervalle de temps. Pour sélectionner les paramètres du graphique, choisissez Réglages. Les données affichées dans le graphique historique peuvent être différentes de celles du graphique de distribution des versions actuelles en raison de la différence entre le nombre de versions de package affichées et également parce que vous pouvez choisir les versions de package à analyser dans le graphique historique. Remarque : Lorsque vous sélectionnez une version de package à visualiser, elle est prise en compte dans le nombre maximum de limites de métriques de flotte. Pour plus d'informations, consultez Limites et quotas d'indexation de la flotte..

Pour une autre méthode permettant de mieux comprendre la collecte de la distribution des versions de packages, voir Collecte de la distribution des versions de packages via `getBucketsAggregation`.

Modèles de requête

L'indexation de flottes avec Software Package Catalog utilise la plupart des fonctionnalités standard prises en charge (telles que les termes, les phrases et les champs de recherche). Cependant, les opérateurs de comparaison (par exemple, inférieur à < et supérieur à>) et les `range` requêtes ne sont pas disponibles pour la `version` clé nommée shadow (`$package`) réservée. Ces requêtes sont disponibles pour la `attributes` clé. Pour plus d'informations, consultez Syntaxe de requête..

Exemple de données

Remarque : pour plus d'informations sur l'ombre nommée réservée et sa structure, consultez Ombre nommée réservée.

Dans cet exemple, un premier périphérique est nommé `AnyThing` et les packages suivants sont installés :

- Package logiciel : `SamplePackage`

  Version du package : `1.0.0`

  Un ID de package : `1111`

L'ombre ressemble à ce qui suit :

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                    }
            }
        }
    }
}
```

Un deuxième périphérique est nommé `AnotherThing` et le package suivant est installé :

- Package logiciel : `SamplePackage`

  Version du package : `1.0.0`

  Un ID de package : `1111`
- Package logiciel : `OtherPackage`

  Version du package : `1.2.5`

  Un ID de package : `2222`

L'ombre ressemble à ce qui suit :

```
{
    "state": {
        "reported": {
            "SamplePackage": {
                "version": "1.0.0",
                "attributes": {
                    "s3UrlForSamplePackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
                    "packageID": "1111"
                }
            },
            "OtherPackage": {
                "version": "1.2.5",
                "attributes": {
                    "s3UrlForOtherPackage": "https://EXAMPIEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile2",
                    "packageID": "2222"
                    }
            },
        }
    }
}
```

Exemples de requêtes

Le tableau suivant répertorie des exemples de requêtes basés sur les exemples d'ombres du périphérique pour `AnyThing` et`AnotherThing`. Pour plus d'informations, consultez Exemples de requêtes.

Dernière version de AWS IoT Device Tester for FreeRTOS

Informations demandées	Interrogation	Result
Éléments sur lesquels une version de package spécifique est installée	`shadow.name.$package.reported.SamplePackage.version:1.0.0`	`AnyThing`, `OtherThing`
Éléments pour lesquels aucune version de package spécifique n'est installée	`NOT shadow.name.$package.reported.OtherPackage.version:1.2.5`	`AnyThing`
Tout appareil utilisant une version de package dont l'ID de package est supérieur à 1500	`shadow.name.$package.reported.*.attributes.packageID>1500"`	`OtherThing`
Éléments sur lesquels un package spécifique est installé et sur lesquels plusieurs packages sont installés	`shadow.name.$package.reported.SamplePackage.version:1.0.0 AND shadow.name.$package.reported.totalCount:2`	`OtherThing`

Collecte de la distribution des versions de packages via **`getBucketsAggregation`**

Outre le panneau Discovery de la AWS IoT console, vous pouvez également obtenir des informations sur la distribution des versions de packages à l'aide de l'opération `GetBucketsAggregation`API. Pour obtenir des informations de distribution de la version du package, vous devez procéder comme suit :

- Définissez un champ personnalisé dans l'indexation de la flotte pour chaque progiciel. Remarque : La création de champs personnalisés est prise en compte dans les AWS IoT quotas du service d'indexation de la flotte.

- Formatez le champ personnalisé comme suit :

```
shadow.name.$package.reported.<packageName>.version
```

Pour plus d'informations, consultez la section Champs personnalisés dans l'indexation des AWS IoT flottes.

Préparation des AWS IoT emplois

AWS IoT Device Management Le catalogue de packages logiciels étend les AWS IoT tâches grâce à des paramètres de substitution et à l'intégration avec l'indexation du AWS IoT parc, les groupes d'objets dynamiques et le nom réservé à l' AWS IoT objet nommé shadow.

> **Note**
>
> Pour utiliser toutes les fonctionnalités proposées par Software Package Catalog, vous devez créer les rôles et politiques AWS Identity and Access Management (IAM) suivants : les droits AWS IoT Jobs pour déployer les versions des packages et les droits AWS IoT Jobs pour mettre à jour le shadow nommé réservé. Pour de plus amples informations, veuillez consulter Preparing security.

Paramètres de substitution pour les AWS IoT tâches

Vous pouvez utiliser des paramètres de substitution comme espace réservé dans votre document de AWS IoT travail. Lorsque le service de tâches rencontre un paramètre de substitution, il pointe la tâche vers l'attribut d'une version logicielle nommée pour la valeur du paramètre. Vous pouvez utiliser ce processus pour créer un document de travail unique et transmettre les métadonnées au travail par le biais d'attributs à usage général. Par exemple, vous pouvez transmettre une URL Amazon Simple Storage Service (Amazon S3), un package logiciel Amazon Resource Name(ARN) ou une signature dans le document de tâche par le biais des attributs de version du package.

Les paramètres de substitution doivent être formatés dans le document de travail comme suit :

- Nom du package logiciel et version du package
  - La chaîne vide entre les deux `package::version` représente le paramètre de substitution du nom du package logiciel. La chaîne vide entre les deux `version::attribute` représente le paramètre de substitution de version du package logiciel. Reportez-vous à l'exemple suivant

pour utiliser le nom du package et les paramètres de substitution de version du package dans un document de travail :. `${aws:iot:package::version::attributes:<attributekey>}`

- Le document de travail remplira automatiquement ces paramètres de substitution à l'aide de l'ARN de version indiqué dans les détails de la version du package. Si vous créez une tâche ou un modèle de tâche pour un déploiement à package unique à l'aide d'une API ou d'une commande CLI, l'ARN de version pour une version de package est représenté par le `destinationPackageVersions` paramètre entre `CreateJob` et`DescribeJob`.

- Tous les attributs d'une version de package logiciel
  - Reportez-vous à l'exemple suivant pour utiliser tous les attributs d'un paramètre de substitution de version de package logiciel dans un document de travail : `${aws:iot:package:<packageName>:version:<versionName>:attributes}`

> **Note**
>
> Le nom du package, la version du package et tous les paramètres de substitution d'attributs peuvent être utilisés ensemble. Reportez-vous à l'exemple suivant pour utiliser les trois paramètres de substitution dans un document de travail : `${aws:iot:package::version::attributes}`

Dans l'exemple suivant, il existe un package logiciel nommé `samplePackage` et sa version de package nommée possède `2.1.5` les attributs suivants :

- nom : `s3URL`, valeur : `https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/exampleCodeFile`
  - Cet attribut identifie l'emplacement du fichier de code stocké dans Amazon S3.
- Nom : `signature`, valeur : `aaaaabbbbbcccccddddeeeeeffffgggggghhhhhiiiiijjjj`
  - Cet attribut fournit une valeur de signature de code dont l'appareil a besoin comme mesure de sécurité. Pour plus d'informations, consultez Code Signing for Jobs. Remarque : Cet attribut est fourni à titre d'exemple et n'est pas obligatoire dans le cadre du catalogue logiciels ou des tâches.

Pours`3URL`, le paramètre du document de travail est écrit comme suit :

```
{
```

```
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
}
```

Pour`signature`, le paramètre du document de travail est écrit comme suit :

```
{
"samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
}
```

Le document de travail complet est rédigé comme suit :

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage":
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
      },
    ],
    "signature": [
      "samplePackage" :
 "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
    ]
  }
}
```

Une fois la substitution effectuée, le document de travail suivant est déployé sur les appareils :

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage": "https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/
exampleCodeFile"
      },
    ],
    "signature": [
      "samplePackage" : "aaaaabbbbbcccccdddddeeeeeffffffgggggghhhhhiiiiijjjj"
```

```
    ]
  }
}
```

Paramètres de substitution (affichage avant et après)

Les paramètres de substitution rationalisent la création d'un document de travail à l'aide de divers indicateurs, par exemple `$default` pour la version du package par défaut. Il n'est donc plus nécessaire de saisir manuellement des métadonnées de version de package spécifiques pour chaque déploiement de tâches, car ces indicateurs sont remplis automatiquement avec les métadonnées référencées dans la version de package spécifique. Pour plus d'informations sur les attributs de version de package, par exemple `$default` pour la version de package par défaut, consultezPréparation du document de travail et de la version du package pour le déploiement.

Dans le AWS Management Console, cliquez sur le bouton Aperçu de la substitution dans la fenêtre de l'éditeur de fichier d'instructions de déploiement lors du déploiement d'une tâche pour une version de package afin d'afficher le document de tâche avec et sans les paramètres de substitution.

À l'aide du paramètre « before-substitution » dans le `DescribeJob` et `GetJobDocument` APIs, vous pouvez afficher la réponse de l'API avant et après la suppression des paramètres de substitution. Reportez-vous aux exemples suivants avec le `DescribeJob` et `GetJobDocument` APIs :

- `DescribeJob`
  - Vue par défaut

    ```
    {
        "jobId": "<jobId>",
        "description": "<description>",
        "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
    TestPackage/version/1.0.2"]
    }
    ```

  - Avant la vue de substitution

    ```
    {
        "jobId": "<jobId>",
        "description": "<description>",
        "destinationPackageVersions": ["arn:aws:iot:us-west-2:123456789012:package/
    TestPackage/version/$default"]
    }
    ```

- `GetJobDocument`
  - Vue par défaut

```
{
    "attributes": {
        "location": "prod-artifacts.s3.us-east-1.amazonaws.com/mqtt-core",
        "signature": "IQoJb3JpZ2luX2VjEIrwEaCXVzLWVhc3QtMSJHMEUCIAofPNPpZ9cI",
        "streamName": "mqtt-core",
        "fileId": "0"
    },
}
```

  - Avant la vue de substitution

```
{
    "attributes": "${aws:iot:package:TestPackage:version:$default:attributes}",
}
```

Pour plus d'informations sur les AWS IoT tâches, la création de documents de tâches et le déploiement de tâches, consultez la section Tâches.

Préparation du document de travail et de la version du package pour le déploiement

Lorsqu'une version de package est créée, son `draft` état indique qu'elle est en cours de préparation pour le déploiement. Pour préparer la version du package en vue du déploiement, vous devez créer un document de tâche, enregistrer le document dans un emplacement accessible à la tâche (tel qu'Amazon S3) et confirmer que la version du package possède les valeurs d'attribut que vous souhaitez que le document de tâche utilise. (Remarque : vous ne pouvez mettre à jour les attributs d'une version de package que lorsqu'elle est dans l'`draft`état.)

Lorsque vous créez un AWS IoT Job ou un modèle de Job pour un déploiement à package unique, vous disposez des options suivantes pour personnaliser votre document de travail :

Fichier d'instructions de déploiement (**`recipe`**)

- Le fichier d'instructions de déploiement pour une version de package contient les instructions de déploiement, y compris un document de travail intégré, pour déployer une version de package sur

plusieurs appareils. Le fichier associe des instructions de déploiement spécifiques à une version de package pour un déploiement rapide et efficace des tâches.

Dans le AWS Management Console, vous pouvez créer le fichier dans la fenêtre d'aperçu du fichier d'instructions de déploiement, dans l'onglet Configurations de déploiement des versions du flux de travail de création d'un nouveau package. Vous pouvez en tirer parti AWS IoT pour générer automatiquement un fichier d'instructions à partir des attributs de la version de votre package en utilisant Démarrer à partir du fichier AWS IoT recommandé ou en utilisant votre document de travail existant stocké dans un compartiment Amazon S3 en utilisant votre propre fichier d'instructions de déploiement.

> **Note**
>
> Si vous utilisez votre propre document de travail, vous pouvez le mettre à jour directement dans la fenêtre d'aperçu du fichier d'instructions de déploiement, mais il ne mettra pas automatiquement à jour votre document de travail d'origine stocké dans votre compartiment Amazon S3.

Lorsque vous utilisez la commande AWS CLI ou une commande d'API telle que `CreatePackageVersion`Get`PackageVersion`, ou`UpdatePackageVersion`, `recipe` représente le fichier d'instructions de déploiement, qui inclut un document de travail intégré.

Pour plus d'informations sur ce qu'est un document de travail, consultezConcepts de base.

Reportez-vous à l'exemple suivant pour le fichier d'instructions de déploiement représenté par `recipe` :

```
{
    "packageName": "sample-package-name",
    "versionName": "sample-package-version",
    ...
    "recipe": "{...}"
}
```

> **Note**
>
> Le fichier d'instructions de déploiement représenté par `recipe` peut être mis à jour lorsqu'une version de package est dans l'`published`état d'état, car il est distinct des métadonnées de version de package. Il devient immuable lors du déploiement des tâches.

`Artifact`attribut de version

- À l'aide de l'attribut `artifact` version de la version de votre package logiciel, vous pouvez ajouter l'emplacement Amazon S3 pour les artefacts de version de votre package. Lorsqu'un déploiement de tâche pour la version de votre package est déclenché à l'aide de AWS IoT Jobs, l'espace réservé `${aws:iot:package:<packageName>:version:<versionName>:artifact-location:s3-presigned-url}` à l'URL présigné dans le document de travail est mis à jour à l'aide du compartiment Amazon S3, de la clé du compartiment et de la version du fichier stocké dans le compartiment Amazon S3. Le compartiment Amazon S3 stockant les artefacts de la version du package doit être situé dans la même région que celle où la version du package a été créée.

> **Note**
>
> Pour stocker plusieurs versions d'objets d'un même fichier dans votre compartiment Amazon S3, vous devez activer le contrôle des versions sur votre compartiment. Pour plus d'informations, consultez la section Activation du versionnement sur les buckets.

Pour accéder aux artefacts de version du package dans le compartiment Amazon S3 lorsque vous utilisez l'opération `CreatePackageVersion` ou `UpdatePackageVersion` API, vous devez disposer des autorisations suivantes :

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion",
```

```
            "Resource": "arn:<partition>:s3:::<bucket>/<key>"
        }
    ]
}
```

Pour plus d'informations sur l'attribut version `artifact` dans les opérations `UpdatePackageVersion` d'API `CreatePackageVersion` et, consultez CreatePackageVersionet UpdatePackageVersion.

Reportez-vous à l'exemple suivant qui montre l'attribut de version `artifact` prenant en charge l'emplacement de l'artefact dans Amazon S3 lors de la création d'une nouvelle version de package :

```
{
    "packageName": "sample package name",
    "versionName": "1.0",
    "artifact": {
        "s3Location": {
            "bucket": "firmware",
            "key": "image.bin",
            "version": "12345"
        }
    }
}
```

> **Note**
>
> Lorsqu'une version de package passe d'un `draft` état d'état à un `published` état de statut, les attributs de version du package et l'emplacement des artefacts deviennent immuables. Pour mettre à jour ces informations, vous devez créer une nouvelle version du package et effectuer ces mises à jour alors que vous êtes dans l'`draft`état d'état.

Version du package

- Une version de package logiciel par défaut peut être indiquée dans les versions disponibles du package logiciel fournissant une version de package sécurisée et stable. Il s'agit de la version de base du package logiciel lors du déploiement de la version par défaut du package sur votre parc d'appareils à l'aide de AWS IoT Jobs. Lorsque vous créez une tâche pour déployer la version

`$default` du package d'un package logiciel, la version du package figurant dans le document de tâche et dans le nouveau déploiement de la tâche doit correspondre à`$default`. La version du package dans le déploiement de la tâche est représentée par `destinationPackageVersions` les commandes d'API et de CLI et `VersionARN` dans le AWS Management Console. La version du package dans le document de tâche est représentée par l'espace réservé de document de tâche suivant, illustré ci-dessous :

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$default
```

Pour créer une tâche ou un modèle de tâche à l'aide de la version du package par défaut, utilisez l'`$default`indicateur dans la commande `CreateJob` ou `CreateJobTemplate` API, comme indiqué ci-dessous :

```
"$ aws iot create-job \
    --destination-package-versions "arn:aws:iot:us-west-2:123456789012:package/
TestPackage/version/$default"
    --document file://jobdoc.json
```

> **Note**
>
> L'attribut de version `$default` du package faisant référence à la version par défaut est un attribut facultatif qui n'est requis que pour référencer la version du package par défaut pour un déploiement de tâche via AWS IoT Jobs.

Lorsque vous êtes satisfait de la version du package, publiez-la via la page de détails du package logiciel dans la AWS IoT console ou en lançant l'opération UpdatePackageVersionAPI. Vous pouvez ensuite référencer la version du package lorsque vous créez la tâche via la AWS IoT console ou en lançant l'opération CreateJobd'API.

Nommer les packages et les versions lors du déploiement

Pour déployer une version de package logiciel sur un appareil, vérifiez que le package logiciel et la version du package référencés dans le document de travail correspondent au package logiciel et à la version du package indiqués dans le `destinationPackageVersions` paramètre de l'opération d'`CreateJob`API. S'ils ne correspondent pas, vous recevrez un message d'erreur vous demandant de faire correspondre les deux références. Pour plus d'informations sur les messages d'erreur du catalogue de packages logiciels, consultezMessages d'erreur généraux relatifs au dépannage.

Outre les packages logiciels et les versions de package référencés dans le document de tâche, vous pouvez inclure des packages logiciels et des versions de package supplémentaires dans le `destinationPackageVersions` paramètre de l'opération d'`CreateJob`API non référencées dans le document de tâche. Assurez-vous que les informations d'installation nécessaires sont incluses dans le document de travail pour que les appareils installent correctement les versions supplémentaires du package logiciel. Pour plus d'informations sur le fonctionnement de l'`CreateJob` API, consultez CreateJob.

Cibler les emplois par le biais de groupes d'objets AWS IoT dynamiques

Le catalogue de logiciels fonctionne avec l'indexation de la flotte, AWS IoT les tâches et les AWS IoT groupes d'objets dynamiques pour filtrer et cibler les appareils de votre flotte afin de sélectionner la version du package à déployer sur vos appareils. Vous pouvez exécuter une requête d'indexation du parc en fonction des informations relatives au package actuel de votre appareil et cibler ces éléments pour une AWS IoT tâche. Vous pouvez également publier des mises à jour logicielles, mais uniquement pour les appareils cibles éligibles. Par exemple, vous pouvez spécifier que vous souhaitez déployer une configuration uniquement sur les appareils qui exécutent actuellement le `iot-device-client 1.5.09`. Pour plus d'informations, consultez Create a dynamic thing group.

Ombre nommée réservée et versions de package

S'il est configuré, les AWS IoT tâches peuvent mettre à jour le nom réservé à un objet nommé shadow (`$package`) lorsque la tâche est terminée avec succès. Dans ce cas, vous n'avez pas besoin d'associer manuellement une version de package à l'Ombre réservée nommée d'un objet.

Vous pouvez choisir d'associer ou de mettre à jour manuellement une version de package à l'ombre nommée réservée à l'objet dans les situations suivantes :

- Vous enregistrez un objet AWS IoT Core sans associer la version du package installé.
- AWS IoT Jobs n'est pas configuré pour mettre à jour le nom réservé à l'objet nommé shadow.
- Vous utilisez un processus interne pour expédier les versions des colis à votre flotte et ce processus n'est pas mis à jour AWS IoT Core une fois terminé.

> **Note**
>
> Nous vous recommandons d'utiliser AWS IoT Jobs pour mettre à jour la version du package dans le nom réservé shadow (`$package`). La mise à jour du paramètre de version dans

l'$packageombre par le biais d'autres processus (tels que des appels d'API manuels ou programmatiques) lorsque AWS IoT Jobs est également configuré pour mettre à jour l'ombre peut entraîner des incohérences entre la version réelle sur l'appareil et la version signalée à l'ombre nommée réservée.

Vous pouvez ajouter ou mettre à jour une version de package sur un objet ombre nommée réservée ($package) via la console ou l'opération UpdateThingShadowAPI. Pour plus d'informations, consultez la section Associer une version de package à un AWS IoT objet.

> **Note**
>
> L'association d'une version de package à un AWS IoT objet ne met pas directement à jour le logiciel de l'appareil. Vous devez déployer la version du package sur l'appareil pour mettre à jour le logiciel de l'appareil.

Désinstallation d'un progiciel et de sa version

$nullest un espace réservé qui invite le service AWS IoT Jobs à supprimer le package logiciel et la version du package existants de l'ombre nommée réservée à l'appareil. $package Pour plus d'informations, veuillez consulter Ombre nommée réservée.

Pour utiliser cette fonctionnalité, remplacez le nom de version à la fin de l'destinationPackageVersionAmazon Resource Name (ARN) par$null. Ensuite, vous devez demander à votre service de supprimer le logiciel de l'appareil.

L'ARN autorisé utilise le format suivant :

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

Par exemple,

```
$ aws iot create-job \
    ... \
    --destinationPackageVersions ["arn:aws:iot:us-east-1:111122223333:package/
samplePackage/version/$null"]
```

Démarrage avec le Catalogue de Logiciels

Vous pouvez créer et gérer le catalogue des packages AWS IoT Device Management logiciels via AWS Management Console les opérations d' AWS IoT Core API et AWS Command Line Interface (AWS CLI).

> **Note**
>
> L'activation AWS IoT de l'indexation du parc est obligatoire pour utiliser le catalogue de packages logiciels. Les opérations de base telles que la création d'une version de package logiciel dans AWS Management Console et l'utilisation de la commande `CreatePackage` API échoueront si l'indexation du AWS IoT parc n'est pas activée.
> Pour plus d'informations sur l'utilisation de l'indexation du AWS IoT parc avec le Software Package Catalog, consultezPréparation de l'indexation de la flotte.

Utilisation de la console

Pour utiliser le AWS Management Console, connectez-vous à votre AWS compte et accédez à AWS IoT Core. Dans le volet de navigation, choisissez Software packages. Vous pouvez ensuite créer et gérer des packages et leurs versions à partir de cette section.

Utilisation des opérations d'API ou de CLI

Vous pouvez utiliser les opérations de l' AWS IoT Core API pour créer et gérer les fonctionnalités du catalogue de packages logiciels. Pour plus d'informations, consultez les AWS IoT sections Référence des API AWS SDKs et boîtes à outils. Les AWS CLI commandes gèrent également votre catalogue. Pour plus d'informations, consultez la AWS IoT Référence de commande CLI;.

Ce chapitre contient les sections suivantes :

- Création d'un package logiciel et d'une version de package
- Déploiement d'une version de package via AWS IoT des tâches
- Associer une version de package à un AWS IoT objet

Création d'un package logiciel et d'une version de package

Vous pouvez utiliser les étapes suivantes pour créer un package et une version initiale via le AWS Management Console.

Pour créer un progiciel

1. Connectez-vous à votre AWS compte et accédez à la AWS IoT console.
2. Dans le panneau de navigation, choisissez Progiciel.
3. Sur la page du AWS IoT package logiciel, choisissez Créer un package. La boîte de dialogue Activer les dépendances pour la gestion des packages s'affiche.
4. Sous Indexation du parc, sélectionnez Ajouter des packages logiciels et une version de l'appareil. Cela est nécessaire pour le catalogue des packages logiciels et fournit une indexation de la flotte et des mesures relatives à votre flotte.
5. [Facultatif] Si vous souhaitez que les AWS IoT tâches mettent à jour l'ombre nommée réservée une fois les tâches terminées avec succès, sélectionnez Mettre à jour automatiquement les ombres des tâches. Si vous ne souhaitez pas que les AWS IoT jobs soient mis à jour, ne cochez pas cette case.
6. [Facultatif] Pour accorder aux AWS IoT jobs le droit de mettre à jour l'ombre nommée réservée, sous Sélectionner un rôle, choisissez Créer un rôle. Si vous ne souhaitez pas que les AWS IoT jobs effectuent cette mise à jour, ce rôle n'est pas obligatoire.
7. Créez ou sélectionnez un rôle.

   a. Si vous n'avez pas de rôle à cette fin : lorsque la boîte de dialogue Créer un rôle apparaît, entrez un nom de rôle, puis choisissez Créer.

   b. Si vous avez un rôle à cette fin : dans Sélectionner un rôle, choisissez votre rôle, puis assurez-vous que la case Attacher la politique au rôle IAM est cochée.
8. Choisissez Confirmer. La page Créer un nouveau package apparaît.
9. Sous Détails du package, entrez un nom de package.
10. Sous Description du package, entrez les informations qui vous aideront à identifier et à gérer ce package.
11. [Facultatif] Vous pouvez utiliser des balises pour vous aider à classer et gérer ce package. Pour ajouter des balises, développez les balises, choisissez la balise Ajouter et entrez une paire clé-valeur. Vous pouvez ajouter jusqu'à 50 balises. Pour plus d'informations, consultez la section Marquage de vos AWS IoT ressources.

Pour ajouter une version de package lors de la création d'un nouveau package

1. Dans Version initiale, entrez un nom de version.

Nous vous recommandons d'utiliser le SemVer format (par exemple,`1.0.0.0`) pour identifier de manière unique la version de votre package. Vous pouvez également utiliser une stratégie de formatage différente mieux adaptée à votre cas d'utilisation. Pour de plus amples informations, veuillez consulter Cycle de vie des versions du package.

2. Sous Description de la version, entrez les informations qui vous aideront à identifier et à gérer cette version du package.

> **Note**
>
> La case à cocher Version par défaut est désactivée car les versions de package sont créées dans un `draft` état. Vous pouvez attribuer un nom à la version par défaut une fois que vous avez créé la version du package et que vous avez modifié l'état en`published`. Pour de plus amples informations, veuillez consulter Cycle de vie des versions du package.

3. [Facultatif] Pour vous aider à gérer cette version ou à communiquer des informations à vos appareils, entrez une ou plusieurs paires nom-valeur pour les attributs de version. Choisissez Ajouter un attribut pour chaque paire nom-valeur que vous entrez. Pour de plus amples informations, veuillez consulter Attributs de version.
4. [Facultatif] Vous pouvez utiliser des balises pour vous aider à classer et gérer ce package. Pour ajouter des balises, développez les balises, choisissez la balise Ajouter et entrez une paire clé-valeur. Vous pouvez ajouter jusqu'à 50 balises. Pour plus d'informations, consultez la section Marquage de vos AWS IoT ressources.
5. Choisissez Suivant.

Associer la nomenclature du logiciel à une version du package (facultatif)

1. À l'étape 3 : Version SBOMs (facultatif), dans la fenêtre des configurations SBOM, choisissez le format de fichier SBOM par défaut et le mode de validation utilisés pour valider votre nomenclature logicielle avant qu'elle ne soit associée à la version de votre package.
2. Dans la fenêtre Ajouter un fichier SBOM, entrez le nom de ressource Amazon (ARN) représentant votre bucket Amazon S3 versionné et le format de fichier SBOM préféré si le type par défaut ne fonctionne pas.

> **Note**
>
> Vous pouvez ajouter un seul fichier SBOM ou un seul fichier zip en contenant plusieurs SBOMs si vous avez plusieurs nomenclatures logicielles pour la version de votre package.

3. Dans la fenêtre Fichier SBOM ajouté, vous pouvez afficher le fichier SBOM que vous avez ajouté pour la version de votre package.
4. Choisissez Créer un package et une version. La page de version du package apparaît et vous pouvez voir l'état de validation de votre fichier SBOM dans la fenêtre du fichier SBOM ajouté. Le statut initial sera celui où `In progress` le fichier SBOM est en cours de validation.

> **Note**
>
> Les statuts de validation des fichiers SBOM sont`Invalid file`,,, `Not started` `In progress` `Validated (SPDX)``Validated (CycloneDX)`, et les raisons de l'échec de la validation.

Déploiement d'une version de package via AWS IoT des tâches

Vous pouvez utiliser les étapes suivantes pour déployer une version de package via le AWS Management Console.

Prérequis :

Avant de commencer, vous devez exécuter les actions suivantes :

- Enregistrez AWS IoT des choses avec AWS IoT Core. Pour savoir comment ajouter vos appareils AWS IoT Core, voir Créer un objet.
- [Facultatif] Créez un AWS IoT groupe d'objets ou un groupe d'objets dynamique pour cibler les appareils sur lesquels vous allez déployer la version du package. Pour savoir comment créer un groupe d'objets, voir Création d'un groupe d'objets statique. Pour savoir comment créer un groupe d'objets dynamique, voir Création d'un groupe d'objets dynamique.
- Créez un package logiciel et une version de package. Pour de plus amples informations, veuillez consulter Création d'un package logiciel et d'une version de package.

- Création d'un document de tâche Pour plus d'informations, consultez Préparation du document de travail et de la version du package pour le déploiement.

Pour déployer une AWS IoT tâche

1. Sur la AWS IoT console, sélectionnez Packages logiciels.
2. Choisissez le package logiciel que vous souhaitez déployer. La page de détails du package logiciel s'affiche.
3. Choisissez la version du package que vous souhaitez déployer, sous Versions, puis choisissez Deploy job version.
4. Si c'est la première fois que vous déployez une tâche via ce portail, une boîte de dialogue décrivant les exigences s'affiche. Vérifiez ces informations, puis choisissez Acknowledge.
5. Entrez un nom pour le déploiement ou laissez le nom généré automatiquement dans le champ Nom.
6. [Facultatif] Dans le champ Description, entrez une description identifiant l'objectif ou le contenu du déploiement, ou laissez les informations générées automatiquement.

   Remarque : Nous vous recommandons de ne pas utiliser d'informations personnellement identifiables dans les champs Nom et description du Job.
7. [Facultatif] Ajoutez les balises à associer à cette tâche.
8. Choisissez Suivant.
9. Sous Objectifs du travail, choisissez les objets ou les groupes d'objets qui devraient recevoir le travail.
10. Dans le champ Fichier de travail, spécifiez le fichier JSON du document de travail.
11. Ouvrez Jobs integration with the Package Catalog service.
12. Sélectionnez les packages et les versions spécifiés dans votre document de travail.

> **Note**
>
> Vous devez choisir les mêmes packages et versions de package que ceux spécifiés dans le document de travail. Vous pouvez en inclure d'autres, mais la tâche émettra des instructions uniquement pour les packages et les versions inclus dans le document de tâche. Pour plus d'informations, voir Nommer les packages et les versions lors du déploiement.

13. Choisissez Suivant.
14. Sur la page de configuration du job, sélectionnez l'un des types de job suivants dans la boîte de dialogue de configuration du job :

    - Tâche de capture instantanée : une tâche de capture instantanée est terminée lorsqu'elle est terminée sur les appareils et les groupes cibles.
    - Tâche continue : une tâche continue s'applique aux groupes d'objets et s'exécute sur tout appareil que vous ajoutez ultérieurement à un groupe cible spécifique.
15. Dans la boîte de dialogue Configurations supplémentaires - facultatives, passez en revue les configurations de tâche facultatives suivantes et effectuez vos sélections en conséquence. Pour plus d'informations, consultez les sections Configurations de déploiement, de planification et d'abandon des tâches et Configurations du délai d'exécution des tâches et des nouvelles tentatives.

    - Configuration du déploiement
    - Planification d’une Configuration.
    - Configuration du délai d'expiration des exécutions de tâches
    - Exécutions de tâches nouvelle tentative de configuration
    - Interruption de la configuration
16. Passez en revue les offres d'emploi, puis choisissez Soumettre.

Une fois que vous avez créé la tâche, la console génère une signature JSON et la place dans votre document de tâche. Vous pouvez utiliser la AWS IoT console pour consulter le statut d'une tâche, annuler ou supprimer une tâche. Pour gérer les tâches, accédez au hub de tâche de la console.

Associer une version de package à un AWS IoT objet

Après avoir installé le logiciel sur votre appareil, vous pouvez associer une version de package à un AWS IoT objet réservé nommé shadow. Si les AWS IoT tâches ont été configurées pour mettre à jour l'ombre nommée réservée à l'objet une fois la tâche déployée et terminée avec succès, vous n'avez pas besoin de suivre cette procédure. Pour de plus amples informations, veuillez consulter Ombre nommée réservée.

Prérequis :

Avant de commencer, vous devez exécuter les actions suivantes :

- Créez un ou AWS IoT plusieurs objets et établissez la télémétrie à travers. AWS IoT Core Pour plus d'informations, consultez la section Mise en route avec AWS IoT Core.
- Créez un package logiciel et une version de package. Pour de plus amples informations, veuillez consulter Création d'un package logiciel et d'une version de package.
- Installez la version logicielle du package sur l'appareil.

> **Note**
>
> L'association d'une version de package à un AWS IoT objet ne met pas à jour ni n'installe le logiciel sur le périphérique physique. La version du package doit être déployée sur l'appareil.

Pour associer une version de package à un AWS IoT objet

1. Surle volet de navigation de la AWS IoT console, développez le menu Tous les appareils et choisissez Objets.
2. Identifiez l' AWS IoT objet que vous souhaitez mettre à jour dans la liste et choisissez le nom de l'objet pour afficher sa page de détails.
3. Dans la section Détails, sélectionnez Packages et versions.
4. Choisissez Ajouter au package et à la version.
5. Pour Choisir un package d'appareil, choisissez le package logiciel de votre choix.
6. Pour Choisir une version, choisissez la version logicielle de votre choix.
7. Choisissez Ajouter un package d'appareil.

   Le package et la version apparaissent dans la liste des packages et versions sélectionnés.
8. Répétez ces étapes pour chaque package et version que vous souhaitez associer à cet élément.
9. Lorsque vous avez terminé, choisissez Ajouter les détails du package et de la version. La page de détails de l'objet s'ouvre et vous pouvez voir le nouveau package et la nouvelle version dans la liste.

AWS IoT Emplois

Utilisez les AWS IoT tâches pour définir un ensemble d'opérations à distance qui peuvent être envoyées et exécutées sur un ou plusieurs appareils connectés AWS IoT. Par exemple, vous pouvez définir une tâche qui ordonne à un ensemble d'appareils de télécharger et d'installer des applications les mises à jour d'un microprogramme, de redémarrer, de procéder à une rotation des certificats ou d'exécuter des opérations de dépannage à distance.

Accès aux AWS IoT offres d'emploi

Vous pouvez commencer à utiliser AWS IoT Jobs à l'aide de la console ou de l' AWS IoT Core API.

Utilisation de la console

Connectez-vous à AWS Management Console, puis accédez à la AWS IoT console. Dans le volet de navigation, choisissez Gérer, puis Tâches. Vous pouvez créer et gérer des tâches depuis cette section. Si vous souhaitez créer et gérer des modèles de tâches, dans le volet de navigation, sélectionnez Modèles de tâches. Pour plus d'informations, veuillez consulter Créez et gérez des tâches à l'aide de AWS Management Console..

Utilisation de l'API ou de la CLI

Vous pouvez commencer en utilisant les opérations de l' AWS IoT Core API. Pour plus d'informations, veuillez consulter AWS IoT Référence d'API . L' AWS IoT Core API sur laquelle les AWS IoT jobs sont basés est prise en charge par le AWS SDK. Pour plus d'informations, voir AWS SDKs et Boîtes à outils.

Vous pouvez utiliser les AWS CLI commandes pour créer et gérer des tâches et des modèles de tâches. Pour plus d'informations, consultez la référence CLI AWS IoT.

AWS IoT Emplois, régions et points de terminaison

AWS IoT Jobs prend en charge les points de terminaison d'API du plan de contrôle et du plan de données spécifiques à votre Région AWS. Les points de terminaison de l'API du plan de données sont spécifiques à votre Compte AWS et Région AWS. Pour plus d'informations sur les points de terminaison AWS IoT des tâches, voir AWS IoT Device Management - points de terminaison des données des tâches dans la référence AWS générale.

Qu'est-ce qu'une opération à distance ?

Une opération à distance est toute mise à jour ou action que vous pouvez effectuer sur un appareil physique, un périphérique virtuel ou un point de terminaison qui peut être effectuée à distance sans la présence physique d'un opérateur ou d'un technicien. L'opération à distance est effectuée à l'aide d'une mise à jour over-the-air (OTA) afin que vos appareils n'aient pas besoin d'être physiquement présents. La gestion de votre parc d'appareils vous AWS Cloud permet d'effectuer des opérations à distance sur vos appareils lorsqu'ils sont enregistrés auprès de AWS IoT Core.

AWS IoT Device Management Jobs propose une approche évolutive pour effectuer des actions à distance sur vos appareils enregistrés auprès de AWS IoT Core. Une tâche est créée dans le AWS Cloud et envoyée à tous les appareils ciblés à l'aide d'une mise à jour OTA via le protocole MQTT ou HTTP.

AWS IoT Device Management Les tâches vous permettent d'effectuer des opérations à distance telles que les réinitialisations d'usine, les redémarrages d'appareils et les mises à jour logicielles OTA de manière sécurisée, évolutive et plus rentable.

Pour plus d'informations sur AWS IoT Core, voirQu'est-ce que c'est AWS IoT ?.

Pour en savoir plus sur la fonction AWS IoT Device Management Jobs, consultez Qu'est-ce que AWS IoT Jobs ?.

Avantages de l'utilisation de AWS IoT Device Management Jobs pour les opérations à distance

L'utilisation de AWS IoT Device Management Jobs pour effectuer vos opérations à distance rationalise la gestion de votre parc d'appareils. La liste suivante met en évidence certains des principaux avantages de l'utilisation de AWS IoT Device Management Tâches pour effectuer vos opérations à distance :

- Intégration parfaite avec d'autres Services AWS
  - AWS IoT Device Management Jobs intègre étroitement la valeur ajoutée Services AWS et les fonctionnalités suivantes :
    - Amazon S3 : stockez vos instructions de fonctionnement à distance dans un compartiment Amazon S3 sécurisé dans lequel vous contrôlez les autorisations d'accès à ce contenu. L'utilisation d'un compartiment Amazon S3 fournit une solution de stockage évolutive et durable qui s'intègre de manière native au catalogue de packages logiciels de gestion des AWS IoT appareils, ce qui permet aux AWS IoT Device Management tâches de faire référence

et de les remplacer dans les instructions de mise à jour. Pour plus d'informations, consultez Qu'est-ce qu'Amazon S ?.

- Amazon CloudWatch : surveillez et enregistrez l'état de mise en œuvre des opérations à distance lors de l'exécution des tâches pour chaque appareil, en plus de l'activité des autres appareils, afin de suivre et d'analyser les performances globales des AWS IoT Device Management tâches dans Jobs. Pour plus d'informations, consultez Qu'est-ce qu'Amazon CloudWatch ? Surveillance des journaux des tâches et capture des données historiques à des fins de résolution des problèmes. Comment ça marche avec jobs.
- Service AWS IoT Device Shadow: Conservez une représentation numérique de votre appareil via AWS IoT un écran invisible à l'aide de AWS IoT Device Management Jobs afin que l'état de votre appareil soit accessible aux applications et aux autres services, quelle que soit la connectivité de l'appareil. Pour de plus amples informations, veuillez consulter Service AWS IoT Device Shadow.
- Fleet Hub pour la gestion des AWS IoT appareils : créez des applications Web autonomes pour surveiller l'état de votre parc d'appareils. Pour plus d'informations, consultez Qu'est-ce que Fleet Hub pour la gestion des AWS IoT appareils ? .

- Bonnes pratiques de sécurité
  - Contrôle des autorisations : contrôlez les autorisations d'accès à vos instructions d'utilisation à distance à l'aide d'Amazon S3 et déterminez quels utilisateurs IAM peuvent déployer vos instructions d'utilisation à distance sur votre parc d'appareils à l'aide de AWS IoT politiques et de rôles d'utilisateur IAM.
    - Pour plus d'informations sur AWS IoT les politiques, consultezCréation d'une AWS IoT politique.
    - Pour plus d'informations, sur les rôles des utilisateurs IAM consultez Gestion des identités et des accès pour AWS IoT.
- Evolutivité
  - Déploiement de tâche ciblé : contrôlez quels appareils reçoivent le document de travail d'une tâche avec un déploiement de tâche ciblé à l'aide de critères de regroupement d'appareils spécifiques saisis dans votre document de tâche lors de la création de la tâche. La création d'un AWS IoT objet pour chaque appareil et le stockage de ces informations dans le AWS IoT registre vous permettent d'effectuer des recherches ciblées à l'aide de l'indexation de flotte. Vous pouvez créer des groupes personnalisés en fonction des résultats de recherche d'indexation du parc afin de soutenir le déploiement de votre projet cible. Pour plus informations, veuillez consulter Gestion des appareils avec AWS IoT. Utilisez les tâches pour effectuer des instantanés plutôt que des tâches continues.

  - État de la tâche : suivez l'état du déploiement du document de travail sur votre parc d'appareils et le statut général de la tâche au niveau du parc d'appareils, en plus de l'état de mise en œuvre individuel du document de travail sur chaque appareil. Pour de plus amples informations, veuillez consulter Tâches et états d'exécution des tâches.
  - Nouvelle évolutivité des appareils : déployez facilement votre document de travail sur un nouvel appareil en l'ajoutant à un groupe personnalisé existant créé à l'aide de l'indexation de la flotte via une tâche continue. Cela vous évitera de devoir déployer le document de travail séparément sur chaque nouvel appareil. Vous pouvez également utiliser une approche plus ciblée avec une capture instantanée en déployant un document de travail sur un groupe prédéterminé d'appareils une fois, puis le travail est terminé.
- Flexibilité
  - Configurations de tâche : personnalisez votre tâche et votre document de travail avec les configurations de tâche facultatives déployées, planifiées, interrompues, temporisées et réessayez pour répondre à vos besoins spécifiques. Pour de plus amples informations, veuillez consulter Configuration de la tâche .
- Rentable
  - Mettez en place une structure de coûts plus efficace pour la maintenance de votre parc d'appareils en tirant parti de AWS IoT Device Management Jobs pour déployer des mises à jour critiques et effectuer des tâches de maintenance de routine. Une solution do-it-yourself (DIY) pour gérer votre parc d'appareils inclut des coûts récurrents et variables tels que l'infrastructure requise pour héberger et gérer la solution de bricolage, les coûts de main-d'œuvre pour développer, maintenir et faire évoluer la solution de bricolage, ainsi que les coûts de transmission de données. En tirant parti de la structure de coûts fixes transparente de AWS IoT Device Management Jobs, vous savez exactement combien coûtera chaque exécution de tâche sur un appareil, en plus des coûts de transmission de données nécessaires pour faciliter le déploiement du document de travail sur votre parc d'appareils et le suivi de l'état d'exécution des tâches pour chaque appareil. Pour en savoir plus, consultez Pricing AWS IoT Core (Tarification).

Qu'est-ce que AWS IoT Jobs ?

Utilisez les AWS IoT tâches pour définir un ensemble d'opérations à distance qui peuvent être envoyées et exécutées sur un ou plusieurs appareils connectés AWS IoT.

Pour créer des tâches, définissez d'abord un document de tâche contenant une liste d'instructions qui décrit les opérations que le périphérique doit effectuer à distance. Pour effectuer ces opérations,

spécifiez une liste de cibles, qui sont des objets individuels, des groupes d'objets ou les deux. Le document de travail et les objectifs constituent ensemble un déploiement.

Chaque déploiement peut comporter des configurations supplémentaires :

- Déploiement : cette configuration définit le nombre d'appareils qui reçoivent le document de travail chaque minute.
- Interrompre : si un certain nombre d'appareils ne reçoivent pas la notification de tâche, utilisez cette configuration pour annuler la tâche. Cela permet d'éviter d'envoyer une mauvaise mise à jour à l'ensemble d'une flotte.
- Délai d'attente : si aucune réponse n'est reçue de la part de vos objectifs d'emploi dans un certain délai, la tâche peut échouer. Vous pouvez suivre le travail en cours d'exécution sur ces appareils.
- Réessayer : si un appareil signale une défaillance ou si une tâche arrive à expiration, vous pouvez utiliser AWS IoT Tâches pour renvoyer automatiquement le document de tâche à l'appareil.
- Planification : cette configuration vous permet de planifier une tâche pour une date et une heure futures. Il vous permet également de créer des fenêtres de maintenance récurrentes qui mettent à jour les appareils pendant des périodes prédéfinies de faible trafic.

AWS IoT Jobs envoie un message pour informer les cibles qu'une tâche est disponible. La cible lance l'exécution de la tâche en téléchargeant le document de tâche, en effectuant les opérations spécifiées et en rendant compte de sa progression AWS IoT. Vous pouvez suivre la progression d'une tâche pour une cible spécifique ou pour toutes les cibles en exécutant les commandes fournies par les AWS IoT tâches. Lorsqu'une tâche démarre, elle a le statut En cours. Les appareils signalent ensuite des mises à jour incrémentielles tout en affichant cet état jusqu'à ce que la tâche réussisse, échoue ou expire.

Les rubriques suivantes décrivent certains concepts clés des tâches ainsi que le cycle de vie des tâches et l'exécution des tâches.

Rubriques

- Concepts clés relatifs aux tâches
- Tâches et états d'exécution des tâches

Concepts clés relatifs aux tâches

Les concepts suivants fournissent des détails sur les AWS IoT tâches et sur la façon de créer et de déployer des tâches pour exécuter des opérations à distance sur vos appareils.

Concepts de base

Les concepts de base que vous devez connaître lorsque vous utilisez AWS IoT Jobs sont les suivants.

Job

Une tâche est une opération distante qui est envoyée vers un ou plusieurs appareils connectés à et exécutée par ceux-ci. AWS IoT Par exemple, vous pouvez définir une tâche qui ordonne à un ensemble d'appareils de télécharger et d'installer une application ou de lancer la mise à jour d'un microprogramme, de redémarrer, de procéder à une rotation des certificats ou d'exécuter des opérations de dépannage à distance.

Document de tâche

Pour créer une tâche, vous devez d'abord créer un document de tâche qui est une description des opérations distantes devant être effectuées par les appareils.

Les documents de tâche sont des documents JSON codés en UTF-8 et ils contiennent l'informations dont vos appareils ont besoin pour effectuer une tâche. Un document de travail contient un ou plusieurs documents URLs dans lesquels l'appareil peut télécharger une mise à jour ou d'autres données. Le document de tâche peut être stocké dans un compartiment Amazon S3 ou être inclus en ligne avec la commande de création de la tâche.

> **Note**
>
> Dans les réponses MQTT, le `jobDocument` champ est un objet JSON. Dans les réponses HTTP, il s'agit d'une représentation sous forme de chaîne de l'objet JSON.

Cible

Lorsque vous créez une tâche, vous spécifiez une liste de cibles qui correspondent aux appareils qui doivent effectuer les opérations. Les cibles peuvent être des objets ou des groupes d'objets, ou les deux. Le service AWS IoT Jobs envoie un message à chaque cible pour l'informer qu'une offre d'emploi est disponible.

Déploiement

Une fois que vous avez créé une tâche en fournissant le document de tâche et en spécifiant votre liste de cibles, le document de tâche est ensuite déployé sur les machines cibles distantes pour lesquelles vous souhaitez effectuer la mise à jour. Pour les tâches de capture instantanée, la tâche sera terminée après le déploiement sur les machines cibles. Pour les tâches continues, une tâche est déployée sur un groupe d'appareils au fur et à mesure qu'ils sont ajoutés aux groupes.

Exécution de tâche

Une exécution de tâche est une instance d'une tâche sur un appareil cible. La cible commence une exécution d'une tâche en téléchargeant le document de tâche. Il exécute ensuite les opérations spécifiées dans le document et rend compte de sa progression à AWS IoT. Un numéro d'exécution est un identifiant unique d'une exécution de tâche sur une cible spécifique. Le service AWS IoT Jobs fournit des commandes permettant de suivre la progression de l'exécution d'une tâche sur une cible et la progression d'une tâche sur toutes les cibles.

Concepts de types de tâche

Les concepts suivants peuvent vous aider à mieux comprendre les différents types de tâches que vous pouvez créer avec AWS IoT Jobs.

Tâche d'instantané

Par défaut, une tâche est envoyée à toutes les cibles que vous spécifiez lorsque vous créez la tâche. Une fois que ces cibles ont terminé la tâche (ou indiqué qu'elles sont dans l'impossibilité de l'exécuter), la tâche est achevée.

Tâche continue

Une tâche continue est envoyée à toutes les cibles que vous spécifiez lorsque vous créez la tâche. Elle continue de s'exécuter et est envoyée à tous les nouveaux appareils (objets) qui sont ajoutées au groupe cible. Par exemple, une tâche continue peut être utilisée pour intégrer ou mettre à niveau des appareils au fur et à mesure qu'ils sont ajoutés à un groupe. Vous pouvez rendre une tâche continue en définissant un paramètre facultatif lors de la création de la tâche.

> **Note**
>
> Lorsque vous ciblez votre parc IoT à l'aide de groupes d'objets dynamiques, nous vous recommandons d'utiliser des tâches continues plutôt que des tâches instantanées. En

utilisant des tâches continues, les appareils qui rejoignent le groupe reçoivent l'exécution de la tâche même après la création de la tâche.

Présigné URLs

Pour un accès sécurisé et limité dans le temps aux données qui ne sont pas incluses dans le document de travail, vous pouvez utiliser Amazon S3 présigné. URLs Placez vos données dans un compartiment et ajouter un lien d'espace réservé aux données du document de tâche. Lorsque AWS IoT Jobs reçoit une demande pour le document de travail, il analyse le document de travail en recherchant les liens réservés, puis remplace les liens par Amazon S3 présigné. URLs

Le lien d'espace réservé a le format suivant :

```
${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

où se *bucket* trouve le nom de votre compartiment et *key* l'objet du compartiment auquel vous créez un lien.

Dans les régions de Pékin et du Ningxia, les œuvres présignées ne URLs fonctionnent que si le propriétaire de la ressource possède une licence ICP (fournisseur de contenu Internet). Pour plus d'informations, consultez Amazon Simple Storage Service dans la documentation Getting Started with AWS Services in China.

Concepts de configuration de tâche

Les concepts suivants peuvent vous aider à comprendre comment configurer les tâches.

Déploiements

Vous pouvez spécifier la vitesse à laquelle les cibles sont averties d'une exécution de tâche en attente. Vous pouvez ainsi créer un déploiement étalé afin de mieux gérer les mises à jour, les redémarrages et autres opérations. Vous pouvez créer une configuration de déploiement en utilisant un taux de déploiement statique ou un taux de déploiement exponentiel. Pour spécifier le nombre maximum d'objectifs de travail à informer par minute, utilisez un taux de déploiement statique.

Pour des exemples de définition des taux de déploiement et pour plus d'informations sur la configuration des déploiements de tâches, consultez Configurations du déploiement, de la planification et de l'annulation des tâches.

Planification

La planification des tâches vous permet de planifier le délai de déploiement d'un document de travail sur tous les appareils du groupe cible pour des tâches continues et instantanées. En outre, vous pouvez créer une fenêtre de maintenance facultative contenant les dates et heures spécifiques auxquelles une tâche déploiera le document de tâche sur tous les appareils du groupe cible. Une fenêtre de maintenance enregistre des cas ayant fréquence quotidiennes, hebdomadaires, mensuelles ou une fréquence de dates et d'heures personnalisées sélectionnées lors de la création initiale de la tâche ou du modèle de tâche. Seules les tâches continues peuvent être planifiées pour effectuer un déploiement pendant une fenêtre de maintenance.

La planification des tâches est spécifique à votre travail. Les Exécutions de Tâches Individuelles ne peuvent pas être planifiées. Pour de plus amples informations, veuillez consulter Configurations du déploiement, de la planification et de l’annulation des tâches.

Interruption

Vous pouvez créer un ensemble de conditions pour interrompre les déploiements lorsque les critères que vous spécifiez sont satisfaits. Pour de plus amples informations, veuillez consulter Configurations du déploiement, de la planification et de l’annulation des tâches.

Délais

Les délais d'expiration de tâche vous permettent de recevoir une notification chaque fois qu'une exécution de tâche se retrouve bloquée dans`IN_PROGRESS` l'état pendant une période étonnamment longue. Il existe deux types de minuteurs : minuteurs d'avancement et minuteurs d'étape. Lorsque la tâche est `IN_PROGRESS`, vous pouvez surveiller et suivre la progression du déploiement de votre tâche.

Les configurations de déploiement et d'interruption sont spécifiques à votre tâche, tandis que la configuration du délai d'expiration est spécifique à un déploiement de tâche. Pour de plus amples informations, veuillez consulter Configurations du délai d’exécution des tâches et des nouvelles tentatives.

Nouvelle tentative

Les nouvelles tentatives permettent de retenter l'exécution d'une tâche lorsqu'une tâche échoue, expire, ou les deux. Vous pouvez avoir jusqu'à 10 tentatives d'exécution de la tâche. Vous pouvez surveiller et suivre la progression de votre nouvelle tentative et déterminer si l'exécution de la tâche a réussi.

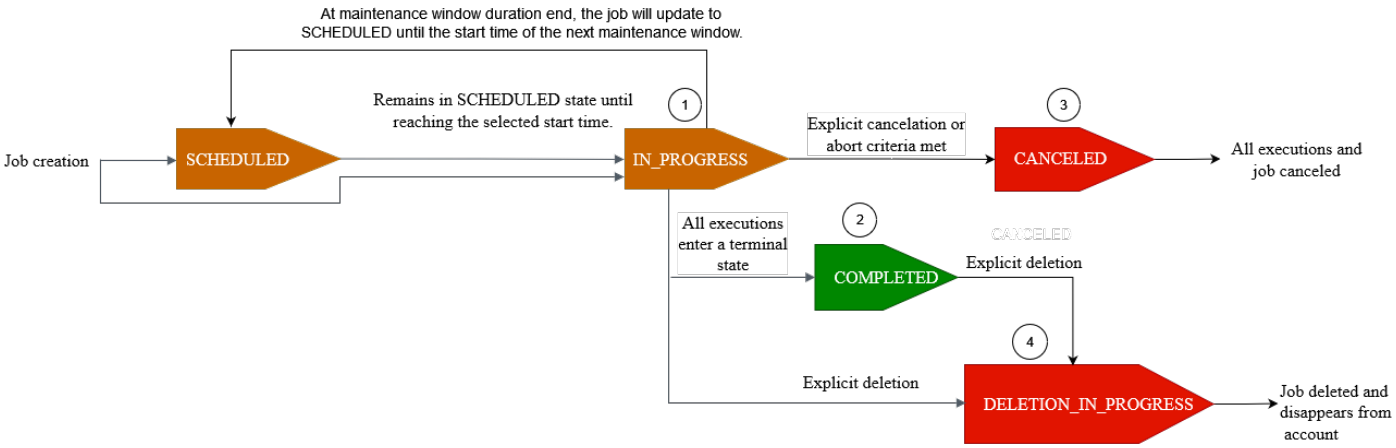
Les configurations de déploiement et d'abandon sont spécifiques à votre tâche, tandis que les configurations de délai d'expiration et de nouvelle tentative sont spécifiques à l'exécution d'une tâche. Pour de plus amples informations, veuillez consulter Configurations du délai d'exécution des tâches et des nouvelles tentatives.

Tâches et états d'exécution des tâches

Les sections suivantes décrivent le cycle de vie d'une AWS IoT tâche et le cycle de vie de son exécution.

États des tâches

Le schéma suivant montre les différents états d'une AWS IoT tâche.



Une tâche que vous créez à l'aide de AWS IoT Jobs peut se trouver dans l'un des états suivants :

- PLANIFIÉ

  Lors de la création initiale de la tâche ou du modèle de tâche à l'aide de la AWS IoT console, CreateJobde CreateJobTemplatel'API ou de l'API, vous pouvez sélectionner la configuration de planification facultative `SchedulingConfig` dans la AWS IoT console ou dans l'CreateJobAPI ou l'CreateJobTemplateAPI. Lorsque vous démarrez une tâche planifiée contenant `unstartTime`, `endTime`, et `endBehavior` spécifique, le statut de la tâche passe à `SCHEDULED`. Lorsque la tâche atteint la fenêtre sélectionnée `startTime` ou la `startTime` fenêtre de maintenance suivante (si vous avez sélectionné le déploiement de la tâche pendant une fenêtre de maintenance), le statut passe de `SCHEDULED` à `IN_PROGRESS` et commence à déployer le document de tâche sur tous les appareils du groupe cible.
- EN_COURS

Lorsque vous créez une tâche à l'aide de la AWS IoT console ou de l'CreateJobAPI, le statut de la tâche passe à`IN_PROGRESS`. Lors de la création, la fonction AWS IoT Jobs, commence à déployer des exécutions de tâches sur les appareils de votre groupe cible. Une fois que toutes les tâches ont été exécutées, la fonction AWS IoT Jobs attend que les appareils exécutent l'action à distance.

Pour plus d'informations sur la simultanéité et les limites applicables aux tâches en cours, consultez AWS IoT Limites d'emplois.

> **Note**
>
> Lorsqu'une `IN_PROGRESS` tâche atteint la fin de la fenêtre de maintenance en cours, le déploiement du document de tâche s'arrête. La tâche sera mise à jour `SCHEDULED` jusqu'à `startTime` la prochaine fenêtre de maintenance.

- TERMINÉ

  Une tâche continue est gérée de l'une des manières suivantes :

  - Pour une tâche continue sans la configuration de planification optionnelle sélectionnée, elle est toujours en cours et continue de fonctionner pour tous les nouveaux appareils ajoutés au groupe cible. Il n'atteindra jamais le statut de`COMPLETED`.
  - Pour une tâche continue avec la configuration de planification facultative sélectionnée, ce qui suit est vrai :
    - Si un `endTime` a été fourni, une tâche continue atteindra le `COMPLETED` statut lorsqu'elle `endTime` sera terminée et que toutes les exécutions de tâches auront atteint le statut terminal.
    - Si un `endTime` n'a pas été fourni dans la configuration de planification facultative, le travail continu continuera à effectuer le déploiement du document de travail.

  Pour une tâche instantanée, le statut de la tâche change en `COMPLETED` lorsque toutes ses exécutions passent à un état terminal, tel que `SUCCEEDED`, `FAILED`, `TIMED_OUT`, `REMOVED`, ou`CANCELED`.

- ANNULÉE

  Lorsque vous annulez une tâche à l'aide de la AWS IoT console, de l'CancelJobAPI ou duConfiguration d’annulation de la tâche , le statut de la tâche passe à`CANCELED`. Lors de l'annulation d'une tâche, AWS IoT Jobs commence à annuler les exécutions de tâches créées précédemment.

Pour plus d'informations sur la simultanéité et les limites applicables aux tâches annulées, consultezAWS IoT Limites d'emplois.

- SUPPRESSION_EN_COURS

  Lorsque vous supprimez une tâche à l'aide de la AWS IoT console ou de l'DeleteJobAPI, le statut de la tâche devientDELETION_IN_PROGRESS. Lors de la suppression d'une tâche, AWS IoT Jobs commence à supprimer les exécutions de tâches créées précédemment. Une fois que toutes les exécutions de tâches ont été supprimées, celles-ci disparaissent de votre AWS compte.

États d'exécution de tâche

Le tableau suivant indique les différents états d'exécution d'une AWS IoT tâche et indique si le changement d'état est initié par le périphérique ou par les AWS IoT tâches.

États et source de l'exécution des tâches

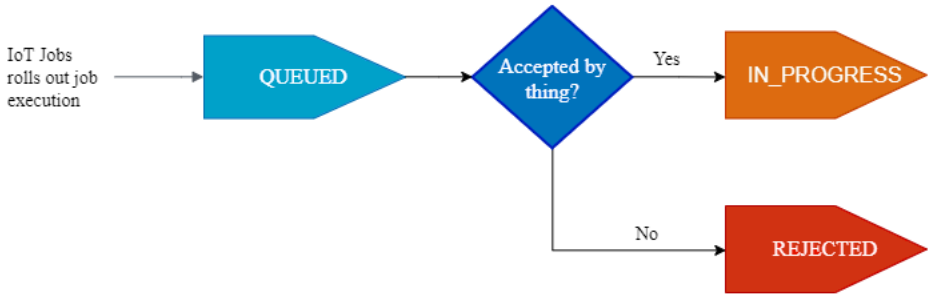
Etat d'exécution de Tâches	Initié par un appareil ?	Initié par AWS IoT Jobs ?	État du terminal ?	Peut être réessayé ?
QUEUED	Non	Oui	Non	Ne s’applique pas
IN_PROGRESS	Oui	Non	Non	Ne s’applique pas
SUCCEEDED	Oui	Non	Oui	Ne s’applique pas
FAILED	Oui	Non	Oui	Oui
TIMED_OUT	Non	Oui	Oui	Oui
REJECTED	Oui	Non	Oui	Non
REMOVED	Non	Oui	Oui	Non
CANCELED	Non	Oui	Oui	Non

La section suivante décrit plus en détail les états d'exécution d'une tâche déployée lorsque vous créez une tâche avec AWS IoT Jobs.

- QUEUED

Lorsque AWS IoT Jobs déploie une exécution de tâche pour un appareil cible, le statut d'exécution de la tâche est défini surQUEUED. L'exécution de la tâche reste au stade QUEUED jusqu'à ce que :

- Votre appareil reçoit l'exécution de la tâche, invoque les opérations API de la fonction Job et indique IN_PROGRESS comme état.
- Vous annulez le travail ou son exécution, ou lorsque les critères d'interruption que vous avez spécifiés sont remplis et que le statut passe à CANCELED.
- Votre appareil est retiré du groupe cible et son statut passe à REMOVED.



- EN_COURS

  Si votre appareil IoT s'abonne à la main réservée Rubriques de tâche $notify et $notify-next que votre appareil invoque l'StartNextPendingJobExecutionAPI ou l'API dont le statut est UpdateJobExecution égal àIN_PROGRESS, AWS IoT Jobs définira le statut d'exécution de la tâche sur. IN_PROGRESS

  L'UpdateJobExecutionAPI peut être invoquée plusieurs fois avec un statut deIN_PROGRESS. Vous pouvez spécifier des détails supplémentaires sur les étapes d'exécution à l'aide de l'statusDetailsobjet.

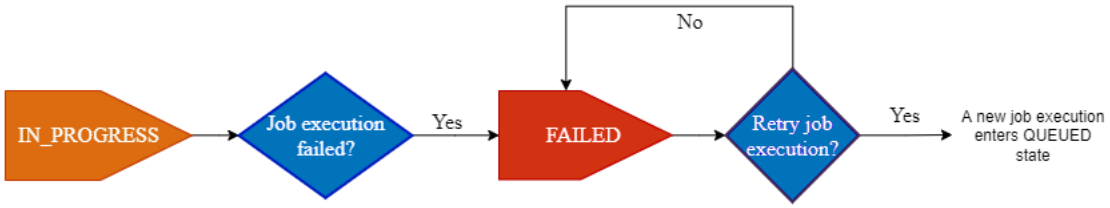
  > **Note**
  >
  > Si vous créez plusieurs tâches pour chaque appareil, les AWS IoT tâches et le protocole MQTT ne garantissent pas l'ordre de livraison.

- RÉUSSI

  Lorsque votre appareil termine avec succès l'opération à distance, il doit appeler l'UpdateJobExecutionAPI avec un statut égal SUCCEEDED à pour indiquer que l'exécution de la tâche a réussi. AWS IoT Jobs est ensuite mis à jour et renvoie le statut d'exécution du travail sous la formeSUCCEEDED.

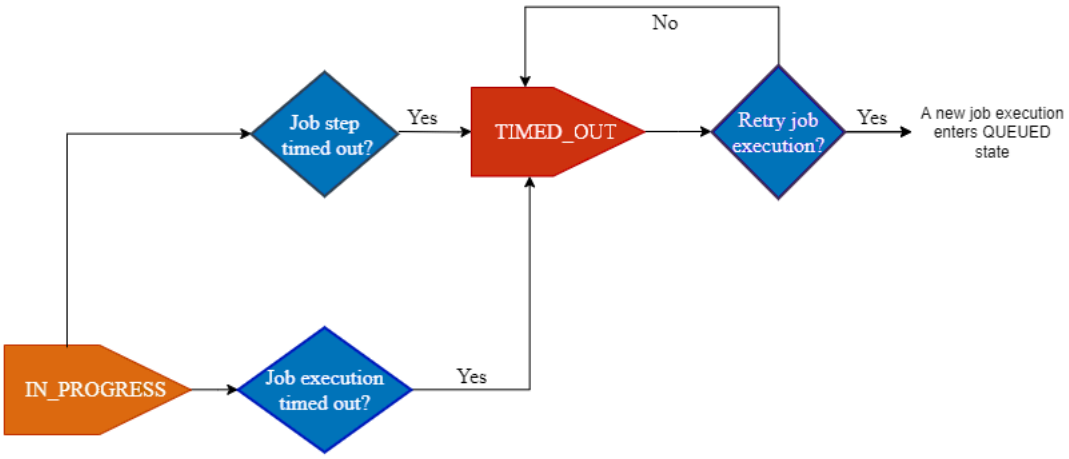
- ÉCHEC

  Lorsque votre appareil ne parvient pas à effectuer l'opération à distance, il doit appeler l'`UpdateJobExecution`API avec un statut égal `Failed` à pour indiquer que l'exécution de la tâche a échoué. AWS IoT Jobs est ensuite mis à jour et renvoie le statut d'exécution du travail sous la forme`Failed`. Vous pouvez réessayer d'exécuter cette tâche pour le périphérique à l'aide de Configuration de nouvelle tentative d’exécution de tâche.



- TIMED_OUT

  Lorsque votre appareil ne parvient pas à terminer une étape de travail alors que le statut est le cas`IN_PROGRESS`, ou lorsqu'il ne parvient pas à terminer l'opération à distance dans le délai imparti, AWS IoT Jobs définit le statut d'exécution de la tâche sur. `TIMED_OUT` Vous disposez également d'un chronomètre pour chaque étape d'une tâche en cours qui s'applique uniquement à l'exécution de la tâche. La durée du temporisateur en cours est spécifiée à l'aide de la `inProgressTimeoutInMinutes` propriété du Configuration du délai d’attente d’exécution de tâche. Vous pouvez réessayer d'exécuter cette tâche pour le périphérique à l'aide du Configuration de nouvelle tentative d’exécution de tâche.



- REFUSÉE

  Lorsque votre appareil reçoit une demande non valide ou incompatible, il doit appeler l'`UpdateJobExecution`API avec un statut de`REJECTED`. AWS IoT Jobs est ensuite mis à jour et renvoie le statut d'exécution du travail sous la forme`REJECTED`.

- SUPPRIMÉ

  Lorsque votre appareil n'est plus une cible valide pour l'exécution de la tâche, par exemple lorsqu'il est détaché d'un groupe d'objets dynamique, la fonction AWS IoT Jobs définit le statut d'exécution de la tâche à `REMOVED`. Vous pouvez rattacher l'objet à votre groupe cible et recommencer l'exécution de la tâche pour l'appareil.

- ANNULÉE

  Lorsque vous annulez un travail ou annulez l'exécution d'un travail à l'aide de la console `CancelJob` ou de l'`CancelJobExecution`API or, ou lorsque les critères d'abandon spécifiés à l'aide du Configuration d’annulation de la tâche sont remplis, AWS IoT Jobs annule le travail et définit le statut d'exécution du travail sur. `CANCELED`

Gestion des tâches

Utilisez des tâches pour informer les appareils d’une mise à jour du logiciel ou du micrologiciel. Vous pouvez utiliser la AWS IoT console, le Opérations d’API de gestion et de contrôle des tâches AWS Command Line Interface, ou le AWS SDKspour créer et gérer des tâches.

Signature de code pour les tâches

Lorsque vous envoyez du code à des appareils, pour que les appareils puissent détecter si le code a été modifié pendant le transport, nous vous recommandons de signer le fichier de code à l'aide du AWS CLI. Pour obtenir des instructions, veuillez consulter Créer et gérer des tâches à l'aide du AWS CLI.

Pour plus d'informations, voir À quoi sert la signature de code AWS IoT ? .

Document de tâche

Avant de créer une tâche, vous devez créer un document de tâche. Si vous utilisez la signature de code pour AWS IoT, vous devez télécharger votre document de travail dans un compartiment Amazon S3 versionné. Pour de plus amples informations sur la création et le chargement d'un fichier dans un compartiment Amazon S3, veuillez consulter Mise en route sur Amazon Simple Storage Service dans le Guide de démarrage Amazon S3.

> **Tip**
>
> Pour des exemples de documents de travail, consultez l'exemple jobs-agent.js dans le AWS IoT SDK pour JavaScript.

Présigné URLs

Votre document de tâche peut contenir une URL Amazon S3 pré-signée qui pointe sur votre fichier de code (ou tout autre fichier). Les Amazon S3 présignés ne URLs sont valides que pour une durée limitée et sont générés lorsqu'un appareil demande un document de travail. Comme l'URL pré-signée n'est pas créée lorsque vous avez créé le document de tâche, vous utilisez une URL d'espace réservé dans votre document de tâche à la place. L'URL d'espace réservé se présente comme suit :

```
${aws:iot:s3-presigned-url-v2:https://
s3.region.amazonaws.com/<bucket>/<code file>}
```

où :

- *bucket*est le compartiment Amazon S3 qui contient le fichier de code.
- *code file*est la clé Amazon S3 du fichier de code.

Lorsqu'un appareil demande le document de travail, AWS IoT génère l'URL présignée et remplace l'URL de remplacement par l'URL présignée. Votre document de tâche est alors envoyé à l'appareil.

Rôle IAM pour autoriser le téléchargement de fichiers depuis S3

Lorsque vous créez une tâche qui utilise Amazon S3 présigné URLs, vous devez fournir un rôle IAM. Le rôle doit octroyer l'autorisation de télécharger les fichiers depuis le compartiment Amazon S3 où les données de tâche/les mises à jour sont stockées. Le rôle doit également accorder à AWS IoT l'autorisation d'endosser le rôle.

Vous pouvez éventuellement spécifier un délai d'expiration pour l'URL présignée. Pour de plus amples informations, veuillez consulter CreateJob.

AWS IoT Autorisez Jobs à assumer votre rôle

1. Allez à la page Centre des rôles de la console IAM et choisissez votre rôle.
2. Dans l'onglet Relations d'approbation, choisissez Modifier la relation d'approbation et remplacez le document de politique par le JSON suivant. Choisissez Mettre à jour la politique d'approbation.

   JSON

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": [
             "iot.amazonaws.com"
           ]
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

3. Ajoutez les clés de condition globale `aws:SourceArn` et `aws:SourceAccount` à la politique pour vous protéger contre le problème de l'adjoint confus.

> **Important**
>
> Votre `aws:SourceArn` doit respecter le format :`arn:aws:iot:`*`region`*`:`*`account-id`*`:*`. Assurez-vous que cela *`region`* correspond à votre AWS IoT région et *`account-id`* à votre numéro de compte client. Pour plus d'informations, consultez Prévention du problème de l'adjoint confus entre services.

```
{
  "Effect": "Allow",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service":
          "iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole",
      "Condition": {
         "StringEquals": {
            "aws:SourceAccount": "123456789012"
         },
         "ArnLike": {
              "aws:SourceArn": "arn:aws:iot:*:123456789012:job/*"
         }
       }
     }
   ]
}
```

4. Si votre travail utilise un document de travail qui est un objet Amazon S3, choisissez Autorisation et utilisez le code JSON suivant. Cela ajoute une politique qui autorise le téléchargement de fichiers depuis votre compartiment Amazon S3 :

   JSON

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

```
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::your_S3_bucket/*"
        }
    ]
}
```

URL présignée pour le téléchargement de fichiers

Si vos appareils doivent télécharger des fichiers dans un compartiment Amazon S3 lors du déploiement d'une tâche, vous pouvez inclure l'espace réservé d'URL présigné suivant dans votre document de travail :

```
${aws:iot:s3-presigned-url-upload:https://s3.region.amazonaws.com/<bucket>/<key>}
```

Vous pouvez utiliser au maximum deux mots de `${thingName}` chaque et `${executionNumber}` comme mots clés réservés dans l'`key`attribut figurant dans l'URL d'espace réservé pour le téléchargement du fichier située dans votre document de travail. `${jobId}` L'espace réservé local représentant ces mots clés réservés dans l'`key`attribut sera analysé et remplacé lors de la création de l'exécution de la tâche. L'utilisation d'un espace réservé local avec des mots clés réservés spécifiques à chaque appareil garantit que chaque fichier téléchargé depuis un appareil est spécifique à cet appareil et qu'il n'est pas remplacé par un fichier similaire téléchargé depuis un autre appareil ciblé par le même déploiement de tâches. Pour plus d'informations sur la résolution des problèmes liés aux espaces réservés locaux dans un espace réservé d'URL présigné pour le téléchargement de fichiers lors du déploiement d'une tâche, consultez. Messages d'erreur généraux relatifs au dépannage

> **Note**
>
> Le nom du compartiment Amazon S3 ne peut pas contenir l'espace réservé local représentant les mots clés réservés pour le fichier chargé. L'espace réservé local doit être situé dans l'`key`attribut.

Cet espace réservé d'URL présignée sera converti en URL de téléchargement présignée Amazon S3 dans votre document de travail lorsqu'un appareil le recevra. Vos appareils l'utiliseront pour télécharger des fichiers vers un compartiment Amazon S3 de destination.

> **Note**
>
> Lorsque le compartiment et la clé Amazon S3 ne sont pas fournis dans l'URL d'espace réservé ci-dessus, AWS IoT Jobs génère automatiquement une clé pour chaque appareil en utilisant un maximum de deux des éléments suivants : `${thingName}${jobId}`, et`${executionNumber}`.

URL présignée à l'aide de la gestion des versions d'Amazon S3

Il est essentiel de protéger l'intégrité d'un fichier stocké dans un compartiment Amazon S3 pour garantir des déploiements de tâches sécurisés en utilisant ce fichier sur votre parc d'appareils. Grâce au versionnement d'Amazon S3, vous pouvez ajouter un identifiant de version pour chaque variante du fichier stockée dans votre compartiment Amazon S3 afin de suivre chaque version du fichier. Cela permet de savoir quelle version du fichier est déployée sur votre parc d'appareils à l'aide de AWS IoT Jobs. Pour plus d'informations sur les compartiments Amazon S3 utilisant le versionnement, consultez Utilisation du versionnement dans les compartiments Amazon S3.

Si le fichier est stocké dans Amazon S3 et que le document de travail contient un espace réservé d'URL présigné, AWS IoT Jobs générera une URL présignée dans le document de travail à l'aide du compartiment Amazon S3, de la clé du compartiment et de la version du fichier stocké dans le compartiment Amazon S3. Cette URL présignée générée dans le document de travail remplacera l'espace réservé d'URL présigné initialement dans le document de travail. Si vous mettez à jour le fichier stocké dans votre compartiment Amazon S3, une nouvelle version du fichier et les versions suivantes `versionId` seront créées pour signaler les mises à jour effectuées et permettre de cibler ce fichier spécifique lors de futurs déploiements de tâches.

Reportez-vous aux exemples suivants pour un aperçu avant et pendant de l'Amazon S3 présigné URLs dans votre document de travail à l'aide du `versionId` :

Espace réservé pour l'URL présignée Amazon S3 (avant le déploiement de Job)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://bucket-name.s3.region-code.amazonaws.com/key-name%3FversionId%3Dversion-id}

//Path-style URL
${aws:iot:s3-presigned-url-v2:https://s3.region-code.amazonaws.com/bucket-name/key-name%3FversionId%3Dversion-id}
```

URL présignée Amazon S3 (pendant le déploiement de la tâche)

```
//Virtual-hosted style URL
${aws:iot:s3-presigned-url-v2:https://sample-bucket-name.s3.us-
west-2.amazonaws.com/sample-code-file.png%3FversionId%3Dversion1}

//Path-style
${aws:iot:s3-presigned-url-v2:https://s3.us-west-2.amazonaws.com/sample-bucket-
name/sample-code-file.png%3FversionId%3Dversion1}
```

Pour plus d'informations sur les objets hébergés virtuellement et de type Path sur Amazon S3 URLs, consultez la section Requêtes et Virtual-hosted-style requêtes de type PATH.

> **Note**
>
> Si vous souhaitez l'ajouter `versionId` à une URL présignée Amazon S3, elle doit être conforme à la prise en charge du codage d'URL. AWS SDK for Java 2.x Pour plus d'informations, consultez Modifications apportées à l'analyse d'Amazon S3 URIs de la version 1 à la version 2.

Différences entre les versions de l'espace réservé aux URL présignées Amazon S3

La liste suivante décrit les différences entre les espaces réservés aux URL présignés Amazon S3 `${aws:iot:s3-presigned-url-v1` (version 1) et `${aws:iot:s3-presigned-url-v2` (version 2) :

- L'espace réservé aux URL présignées Amazon S3 n'est `${aws:iot:s3-presigned-url-v1` pas pris en charge. `version-id`
- L'espace réservé pour l'URL présignée Amazon S3 `${aws:iot:s3-presigned-url-v1` reçoit l'URL Amazon S3 non codée. L'espace réservé à l'URL présignée Amazon S3 `${aws:iot:s3-presigned-url-v2` nécessite que l'URL Amazon S3 soit codée conformément à la norme du SDK Amazon S3.

Rubriques

- Créez et gérez des tâches à l'aide de AWS Management Console.
- Créez et gérez des emplois à l'aide du AWS CLI

Créez et gérez des tâches à l’aide de AWS Management Console.

Cette section décrit comment créer et gérer des tâches à partir de la AWS IoT console. Après avoir créé un travail, vous pouvez consulter les informations le concernant sur la page de détails et gérer le travail.

> **Note**
>
> Si vous souhaitez effectuer une signature de code pour AWS IoT des tâches, utilisez le AWS CLI. Pour plus d'informations, voir Créer et gérer des tâches à l'aide du AWS CLI.

Rubriques

- Créez et gérez des tâches à l'aide du AWS Management Console
- Consultez et gérez les tâches à l'aide du AWS Management Console

Créez et gérez des tâches à l'aide du AWS Management Console

Pour créer une tâche, connectez-vous à la AWS IoT console et accédez au hub de tâches dans la section Actions à distance. Effectuez ensuite les étapes suivantes.

1. Sur la page Tâches, dans la boîte de dialogue Tâches, choisissez Créer une tâche.
2. En fonction de l'appareil que vous utilisez, vous pouvez créer une tâche personnalisée, une tâche de mise à jour FreeRTOS OTA ou une tâche. AWS IoT Greengrass Pour cet exemple, choisissez Créer une tâche personnalisée. Choisissez Suivant.
3. Sur la page Propriétés de la tâche personnalisée, dans la boîte de dialogue Propriétés de la tâche, entrez vos informations pour les champs suivants :

   - Nom : Entrez un nom de tâche alphanumérique unique.
   - Description - facultatif : Entrez une description facultative de votre tâche.
   - Balises — en option :

   > **Note**
   >
   > Nous vous recommandons de ne pas utiliser d'informations personnelles identifiables dans votre poste IDs et dans votre description.

Choisissez Suivant.

4. Sur la page Configuration du fichier de la boîte de dialogue Cibles du travail, sélectionnez les objets ou les groupes d'objets que vous souhaitez exécuter.

   Dans la boîte de dialogue Document de tâche, choisissez l'une des options suivantes :

   - Depuis le fichier : fichier de tâche JSON que vous avez chargé dans un compartiment Amazon S3
     - Signature de code

       Dans le document de travail situé dans votre URL Amazon S3, `${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}` est requis comme espace réservé jusqu'à ce qu'il soit remplacé par le chemin du fichier de code signé à l'aide de votre profil de signature de code. Le nouveau fichier de code signé apparaîtra initialement dans un dossier `SignedImages` de votre compartiment source Amazon S3. Un nouveau document de travail contenant un préfixe `Codesigned_` sera créé avec le chemin du fichier de code signé remplaçant l'espace réservé au signe de code et placé dans votre URL Amazon S3 pour créer une nouvelle tâche.
     - Ressource de pré-signature URLs

       Dans le menu déroulant Rôle de pré-signature, choisissez le rôle IAM que vous avez créé dans Presigned. URLs L'utilisation `${aws:iot:s3-presigned-url:` URLs pour présigner des objets situés dans Amazon S3 est une bonne pratique de sécurité pour les appareils qui téléchargent des objets depuis Amazon S3.

       Si vous souhaitez utiliser Presigned comme espace réservé URLs pour la signature de code, utilisez l'exemple de modèle suivant :

       ```
       ${aws:iot:s3-presigned-url:${aws:iot:code-sign-signature:<S3 URL>}
       ```

   - À partir du modèle : modèle de tâche contenant un document de tâche et des configurations de tâche. Le modèle de tâche peut être un modèle de tâche personnalisé que vous avez créé ou un modèle AWS géré.

     Si vous créez une tâche pour effectuer des actions à distance fréquemment utilisées, telles que le redémarrage de votre appareil, vous pouvez utiliser un modèle AWS géré. Ces modèles ont déjà été préconfigurés pour être utilisés. Pour plus d'informations, consultez Créer un

modèle de tâche personnalisé et Créez des modèles de tâches personnalisés à partir de modèles gérés.

5. Sur la page Configuration des tâches de la boîte de dialogue Configuration des tâches, sélectionnez l'un des types de tâches suivants :

   - Tâche de capture instantanée : une tâche de capture instantanée est terminée lorsqu'elle est terminée sur les appareils et les groupes cibles.
   - Tâche continue : une tâche continue s'applique aux groupes d'objets et s'exécute sur tout appareil que vous ajoutez ultérieurement à un groupe cible spécifique.

6. Dans la boîte de dialogue Configurations supplémentaires - facultatif, passez en revue les configurations de tâches facultatives suivantes et effectuez vos sélections en conséquence :

   - Configuration du déploiement
   - Configuration de la planification
   - Configuration du délai d'expiration des exécutions de tâches
   - Configuration des nouvelles tentatives d'exécution des tâches - nouveau
   - Abandonner la configuration

   Reportez-vous aux sections suivantes pour plus d'informations sur les configurations de tâches :

   - Configurations du déploiement, de la planification et de l'annulation des tâches
   - Configurations du délai d'exécution des tâches et des nouvelles tentatives

   Passez en revue toutes vos sélections d'emplois, puis choisissez Soumettre pour créer votre emploi.

Consultez et gérez les tâches à l'aide du AWS Management Console

Une fois que vous avez créé la tâche, la console génère une signature JSON et la place dans votre document de tâche. Vous pouvez utiliser la console AWS IoT pour afficher le statut d'une tâche, annuler une tâche ou la supprimer.

Si vous choisissez le job que vous avez créé, vous pouvez trouver :

- Détails généraux de la tâche, tels que le nom, la description, le type, l'heure à laquelle elle a été créée, sa dernière mise à jour et l'heure de début estimée.

- Toutes les configurations de tâches que vous avez spécifiées et leur statut.
- Document de tâche.
- Les exécutions des tâches et toutes les balises facultatives que vous avez spécifiées.

Pour gérer les tâches, accédez au hub de tâches de la console et choisissez si vous souhaitez modifier, supprimer ou annuler la tâche.

Créez et gérez des emplois à l'aide du AWS CLI

Cette section décrit comment créer et gérer des tâches.

Création de tâches

Pour créer une AWS IoT tâche, utilisez la CreateJob commande. La tâche est mise en file d'attente en vue de son exécution sur les cibles (objets ou groupes d'objets) que vous spécifiez. Pour créer une AWS IoT tâche, vous avez besoin d'un document de tâche qui peut être inclus dans le corps de la demande ou sous forme de lien vers un document Amazon S3. Si la tâche inclut le téléchargement de fichiers à l'aide d'Amazon S3 présigné URLs, vous avez besoin d'un rôle IAM Amazon Resource Name (ARN) autorisé à télécharger le fichier et autorisant le service AWS IoT Jobs à assumer ce rôle.

Pour plus d'informations sur la syntaxe lors de la saisie de la date et de l'heure à l'aide d'une commande API ou du AWS CLI, consultez Timestamp.

Signature de code avec les tâches

Si vous utilisez la signature de code pour AWS IoT, vous devez démarrer une tâche de signature de code et inclure le résultat dans votre document de tâche. Cela remplacera l’espace réservé à la signature du signe de code dans votre document de tâche, qui est requis comme espace réservé jusqu’à ce qu’il soit remplacé par le chemin du fichier de code signé à l’aide de votre profil de signature de code. L’espace réservé pour la signature du code se présente comme suit :

```
${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}
```

Utilisez la start-signing-jobcommande pour créer une tâche de signature de code. `start-signing-job`brenvoie un identifiant de tâche. Utilisez la commande describe-signing-job pour obtenir l’emplacement Amazon S3 dans lequel la signature est stockée. Vous pouvez alors télécharger la signature depuis Amazon S3. Pour en savoir plus sur les tâches de signature de code, consultez Signature de code pour AWS IoT.

Votre document de tâche doit contenir un espace réservé d'URL pré-signée pour votre fichier de code et la sortie de la signature JSON être placée dans un compartiment Amazon S3 à l'aide de la commande start-signing-job :

```
{
    "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/
image}",
}
```

Créer une tâche avec un document de tâche

La commande suivante montre comment créer une tâche à l'aide d'un document de tâche (*job-document.json*) stocké dans un compartiment Amazon S3 (*jobBucket*) et d'un rôle autorisé à télécharger des fichiers depuis Amazon S3 (*S3DownloadRole*).

```
aws iot create-job  \
      --job-id 010  \
      --targets arn:aws:iot:us-east-1:123456789012:thing/thingOne  \
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute
\": 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings
\": 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

La tâche est exécutée*thingOne*.

Le paramètre facultatif `timeout-config` spécifie la durée allouée à chaque appareil pour terminer l'exécution de la tâche. Le minuteur est démarré quand l'état de l'exécution de la tâche a la valeur `IN_PROGRESS`. Si l'état de l'exécution de la tâche n'est pas défini sur un autre état terminal avant l'expiration, il est défini avec la valeur `TIMED_OUT`.

Le minuteur en cours ne peut pas être mis à jour et s'applique à toutes les exécutions de tâche pour la tâche. Chaque fois qu'une exécution de tâche reste dans cet `IN_PROGRESS` état pendant plus

longtemps que cet intervalle, elle échoue et passe à l'`TIMED_OUT`état du terminal. AWS IoT publie également une notification MQTT.

Pour plus d'informations sur la création de configurations relatives aux déploiements et interruptions de tâche, consultez la section Configuration du déploiement et de l'interruption des tâches.

> **Note**
>
> Les documents de tâche qui sont spécifiés en tant que fichiers Amazon S3 sont extraits au moment de la création de la tâche. La modification du contenu du fichier Amazon S3 que vous avez utilisé comme source de votre document de tâche une fois que vous avez créé la tâche ne modifie pas ce qui est envoyé aux cibles de la tâche.

Mettre à jour une tâche

Vous utilisez la commande UpdateJob pour mettre à jour une tâche. Vous pouvez mettre à jour les champs `description`, `presignedUrlConfig`, `jobExecutionsRolloutConfig`, `abortConfig` et `timeoutConfig` d'une tâche.

```
aws iot update-job  \
  --job-id 010  \
  --description "updated description" \
  --timeout-config inProgressTimeoutInMinutes=100 \
  --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\": 50,
 \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
 \"numberOfSucceededThings\": 1000}, \"maximumPerMinute\": 1000}}" \
  --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
  --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

Pour en savoir plus, consultez Configuration du déploiement et de l'interruption des tâches.

Annuler une tâche

Vous utilisez la commande CancelJob pour annuler une tâche. L'annulation d'une tâche AWS IoT empêche le déploiement de toute nouvelle exécution de tâche pour la tâche. Il annule également toutes les exécutions de tâches effectuées dans un `QUEUED` État. AWS IoT conserve intacte toute

exécution de tâche dans un état terminal car le périphérique a déjà terminé la tâche. Si une exécution de tâche a le statut `IN_PROGRESS`, elle reste aussi en l'état, à moins que vous utilisiez le paramètre facultatif `--force`.

La commande suivante montre comment annuler une tâche avec l'ID 010.

```
aws iot cancel-job --job-id 010
```

La commande affiche la sortie suivante :

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

Lorsque vous annulez une tâche, les exécutions de tâche dont l'état est `QUEUED` sont annulées. Les exécutions de tâche dont l'état est `IN_PROGRESS` sont annulées si vous spécifiez le paramètre facultatif `--force`. Les exécutions de tâche qui se trouvent dans un état terminal ne sont pas annulées.

> **Warning**
>
> L'annulation d'une tâche dont l'état est `IN_PROGRESS` (en définissant le paramètre `--force`) a pour effet d'annuler les exécutions de tâche en cours et empêche l'appareil qui exécute la tâche de mettre à jour le statut d'exécution de la tâche. Soyez vigilant et vérifiez que chaque appareil exécutant une tâche annulée est en mesure de reprendre un état valide.

Le statut d'une tâche annulée ou de l'une de ses exécutions de tâches est finalement cohérent. AWS IoT arrête de planifier les nouvelles exécutions de `QUEUED` tâches et les exécutions de tâches pour cette tâche sur les appareils dès que possible. La modification de l'état de l'exécution d'une tâche en `CANCELED` peut prendre du temps, selon le nombre d'appareils et autres facteurs.

Si une tâche est annulée, car elle a satisfait les critères définis par un objet `AbortConfig`, le service ajoute les valeurs renseignées automatiquement pour les champs `comment` et `reasonCode`. Vous pouvez créer vos propres valeurs pour `reasonCode` lorsque la tâche d'annulation est orientée utilisateurs.

Annulation d'une exécution de tâche

La commande CancelJobExecution permet d'annuler une exécution de tâche sur un appareil déterminé. Elle annule l'exécution de la tâche qui se trouve dans un état `QUEUED`. Si vous souhaitez annuler une exécution de tâche dont le statut est en cours, vous devez utiliser le paramètre `--force`.

La commande suivante montre comment annuler l'exécution d'une tâche depuis la tâche 010 s'exécutant sur `myThing`.

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

La commande n'affiche aucune sortie.

Une exécution de tâche qui se trouve dans un état `QUEUED` est annulée. Une exécution de tâche dont l'état est `IN_PROGRESS` est annulée mais si vous spécifiez le paramètre facultatif `--force`. Les exécutions de tâche qui se trouvent dans un état terminal ne peuvent pas être annulées.

> **Warning**
>
> Lorsque vous annulez une exécution de tâche dont le statut est `IN_PROGRESS`, l'appareil ne peut pas mettre à jour le statut d'exécution de la tâche. Soyez vigilant et vérifiez que l'appareil est en mesure de reprendre un état valide.

Si l'exécution de tâche se trouve dans un état terminal ou si l'exécution de tâche présente le statut `IN_PROGRESS` et que le paramètre `--force` a la valeur `true`, cette commande entraîne une exception `InvalidStateTransitionException`.

Le statut d'une exécution de tâche annulée est cohérent à terme. La modification de l'état de l'exécution d'une tâche en `CANCELED` peut prendre du temps, selon différents facteurs.

Suppression d'une tâche

La commande DeleteJob permet de supprimer une tâche et les exécutions de tâche associées. Par défaut, vous pouvez uniquement supprimer une tâche qui se trouve dans un état terminal (`SUCCEEDED` ou `CANCELED`). Dans le cas contraire, une exception se produit. Vous ne pouvez supprimer une tâche qui se trouve dans l'état `IN_PROGRESS` que si le paramètre `force` a la valeur `true`.

Pour supprimer une tâche, exécutez la commande suivante :

```
aws iot delete-job --job-id 010 --force|--no-force
```

La commande n'affiche aucune sortie.

> **Warning**
>
> Lorsque vous supprimez une tâche qui est dans l'état `IN_PROGRESS`, le périphérique qui déploie la tâche ne peut pas accéder aux informations sur la tâche ou mettre à jour le statut d'exécution de la tâche. Soyez vigilant et vérifiez que chaque appareil déployant une tâche annulée est en mesure de reprendre un état valide.

La suppression d'une tâche peut prendre un certain temps qui varie en fonction du nombre d'exécutions de tâche créées pour la tâche et autres facteurs. Pendant la suppression de la tâche, l'état de celle-ci indique `DELETION_IN_PROGRESS`. Toute tentative de suppression ou d'annulation d'une tâche dont le statut est `DELETION_IN_PROGRESS` entraîne une erreur.

Seules 10 tâches peuvent avoir l'état `DELETION_IN_PROGRESS` en même temps. Sinon, une exception `LimitExceededException` se produit.

Obtention d'un document de tâche

Vous utilisez la commande GetJobDocument pour récupérer un document de tâche pour une tâche. Un document de tâche est une description des opérations distantes à exécuter par les appareils.

Exécutez la commande suivante pour obtenir un document de tâche.

```
aws iot get-job-document --job-id 010
```

La commande renvoie le document de tâche de la tâche spécifiée :

```
{
    "document": "{\n\t\"operation\":\"install\",\n\t\"url\":\"http://amazon.com/
firmWareUpate-01\",\n\t\"data\":\"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
amzn-s3-demo-bucket/datafile}\"\n}"
}
```

> **Note**
>
> Lorsque vous utilisez cette commande pour récupérer un document de travail, l'espace réservé n'est URLs pas remplacé par Amazon S3 présigné. URLs Lorsqu'un appareil appelle l'opération GetPendingJobExecutionsAPI, l'espace réservé URLs est remplacé par Amazon S3 présigné URLs dans le document de travail.

Affichage des tâches

Pour obtenir la liste de toutes les tâches de votre Compte AWS choix, utilisez la ListJobs commande. Les données de travail et d'exécution de travail sont conservées pendant une durée limitée. Exécutez la commande suivante pour répertorier toutes les tâches de votre Compte AWS :

```
aws iot list-jobs
```

La commande répertorie toutes les tâches de votre compte, triées sur le statut de la tâche :

```
{
    "jobs": [
        {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486687079.743,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/013",
            "createdAt": 1486687079.743,
            "targetSelection": "SNAPSHOT",
            "jobId": "013"
        },
        {
            "status": "SUCCEEDED",
            "lastUpdatedAt": 1486685868.444,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/012",
            "createdAt": 1486685868.444,
            "completedAt": 148668789.690,
            "targetSelection": "SNAPSHOT",
            "jobId": "012"
        },
        {
            "status": "CANCELED",
            "lastUpdatedAt": 1486678850.575,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/011",
```

```
            "createdAt": 1486678850.575,
            "targetSelection": "SNAPSHOT",
            "jobId": "011"
        }
    ]
}
```

Description d'une tâche

La commande DescribeJob permet d'obtenir le statut d'une tâche. La commande suivante explique comment décrire une tâche :

```
$ aws iot describe-job --job-id 010
```

La commande renvoie le statut de la tâche spécifiée. Par exemple :

```
{
    "documentSource": "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-
document.json",
    "job": {
        "status": "IN_PROGRESS",
        "jobArn": "arn:aws:iot:us-east-1:123456789012:job/010",
        "targets": [
            "arn:aws:iot:us-east-1:123456789012:thing/myThing"
        ],
        "jobProcessDetails": {
            "numberOfCanceledThings": 0,
            "numberOfFailedThings": 0,
            "numberOfInProgressThings": 0,
            "numberOfQueuedThings": 0,
            "numberOfRejectedThings": 0,
            "numberOfRemovedThings": 0,
            "numberOfSucceededThings": 0,
            "numberOfTimedOutThings": 0,
            "processingTargets": [
                arn:aws:iot:us-east-1:123456789012:thing/thingOne,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupOne,
                arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupTwo
            ]
        },
        "presignedUrlConfig": {
            "expiresInSec": 60,
```

```
            "roleArn": "arn:aws:iam::123456789012:role/S3DownloadRole"
        },
        "jobId": "010",
        "lastUpdatedAt": 1486593195.006,
        "createdAt": 1486593195.006,
        "targetSelection": "SNAPSHOT",
        "jobExecutionsRolloutConfig": {
            "exponentialRate": {
                "baseRatePerMinute": integer,
                "incrementFactor": integer,
                "rateIncreaseCriteria": {
                    "numberOfNotifiedThings": integer, // Set one or the other
                    "numberOfSucceededThings": integer // of these two values.
                },
            "maximumPerMinute": integer
         }
        },
        "abortConfig": {
            "criteriaList": [
                {
                    "action": "string",
                    "failureType": "string",
                    "minNumberOfExecutedThings": integer,
                    "thresholdPercentage": integer
                }
            ]
        },
        "timeoutConfig": {
           "inProgressTimeoutInMinutes": number
          }
    }
}
```

Affichage des exécutions d'une tâche

Une tâche en cours d'exécution sur un appareil spécifique est représentée par un objet d'exécution de tâche. La commande ListJobExecutionsForJob permet d'afficher toutes les exécutions de tâche d'une tâche. L'exemple suivant montre comment afficher les exécutions d'une tâche :

```
aws iot list-job-executions-for-job --job-id 010
```

La commande renvoie une liste d'exécutions de tâche :

```
{
    "executionSummaries": [
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 1234567890
        }
    },
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingTwo",
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486593345.659,
            "queuedAt": 1486593196.378,
            "startedAt": 1486593345.659,
            "executionNumber": 4567890123
        }
    }
    ]
}
```


Affichage des exécutions de tâche pour un objet

La commande ListJobExecutionsForThing permet d'afficher toutes les exécutions de tâche s'exécutant sur un objet. L'exemple suivant montre comment afficher les exécutions de tâche d'un objet :

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

La commande renvoie la liste des exécutions de tâche qui sont en cours d'exécution ou qui se sont exécutées sur l'objet spécifié :


```
{
    "executionSummaries": [
    {
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486687082.071,
```

```
            "queuedAt": 1486687082.071,
            "executionNumber": 9876543210
        },
        "jobId": "013"
    },
    {
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "startAt": 1486685870.729,
            "lastUpdatedAt": 1486685870.729,
            "queuedAt": 1486685870.729,
            "executionNumber": 1357924680
        },
        "jobId": "012"
    },
    {
        "jobExecutionSummary": {
            "status": "SUCCEEDED",
            "startAt": 1486678853.415,
            "lastUpdatedAt": 1486678853.415,
            "queuedAt": 1486678853.415,
            "executionNumber": 4357680912
        },
        "jobId": "011"
    },
    {
        "jobExecutionSummary": {
            "status": "CANCELED",
            "startAt": 1486593196.378,
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 2143174250
        },
        "jobId": "010"
    }
    ]
}
```

Description d'une exécution de tâche

La commande DescribeJobExecution permet d'obtenir le statut d'une exécution de tâche. Pour identifier l'exécution de tâche, vous devez spécifier un ID de tâche, un nom d'objet et éventuellement un numéro d'exécution. La commande suivante explique comment décrire une exécution de tâche :

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

La commande renvoie la chaîne JobExecution. Par exemple :

```
{
    "execution": {
        "jobId": "017",
        "executionNumber": 4516820379,
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "versionNumber": 123,
        "createdAt": 1489084805.285,
        "lastUpdatedAt": 1489086279.937,
        "startedAt": 1489086279.937,
        "status": "IN_PROGRESS",
        "approximateSecondsBeforeTimedOut": 100,
        "statusDetails": {
            "status": "IN_PROGRESS",
            "detailsMap": {
                "percentComplete": "10"
            }
        }
    }
}
```

Suppression d'une exécution de tâche

Pour supprimer une exécution de tâche, exécutez la commande DeleteJobExecution. Pour identifier l'exécution de tâche, vous devez spécifier un ID de tâche, un nom d'objet et un numéro d'exécution. La commande suivante explique comment supprimer une exécution de tâche :

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
 1234567890 --force|--no-force
```

La commande n'affiche aucune sortie.

Par défaut, le statut de l’exécution d’une tâche doit être `QUEUED` ou dans un état terminal (`SUCCEEDED`, `FAILED`, `REJECTED`, `TIMED_OUT`, `REMOVED` ou `CANCELED`). Dans le cas contraire, une erreur se produit. Pour supprimer une exécution de tâche avec le statut `IN_PROGRESS`, vous pouvez définir le paramètre `force` sur `true`.

> **Warning**
>
> Lorsque vous supprimez une exécution de tâche qui est dans l'état `IN_PROGRESS`, le périphérique qui exécute la tâche ne peut pas accéder aux informations sur la tâche ou mettre à jour le statut d'exécution de la tâche. Soyez vigilant et vérifiez que l'appareil est en mesure de reprendre un état valide.

Modèles de tâche

Utilisez des modèles de tâches pour préconfigurer des tâches que vous pouvez déployer sur plusieurs ensembles de machines cibles. Pour déployer des actions à distance fréquemment effectuées sur vos appareils, telles que le redémarrage ou l'installation d'une application, vous pouvez utiliser des modèles pour définir des configurations standard. Pour effectuer des opérations telles que le déploiement de correctifs de sécurité et de corrections de bogues, vous pouvez créer des modèles à partir de tâches existantes.

Lorsque vous créez un modèle de tâche, spécifiez les configurations et ressources supplémentaires suivantes.

- Propriétés de la tâche
- Documents de travail et objectifs
- Critères de déploiement, de planification et d'annulation
- Délai d'expiration et critères de nouvelle tentative

Modèles personnalisés et AWS gérés

En fonction de l'action à distance que vous souhaitez effectuer, vous pouvez créer un modèle de tâche personnalisé ou utiliser un modèle AWS géré. Utilisez des modèles de tâches personnalisés pour fournir votre propre document de travail personnalisé et créer des tâches réutilisables à déployer sur vos appareils. AWS les modèles gérés sont des modèles de tâches fournis par AWS IoT Jobs pour les actions fréquemment effectuées. Ces modèles comportent un document de tâche prédéfini pour certaines actions à distance, de sorte que vous n'avez pas à créer votre propre document de tâche. Les modèles gérés vous aident à créer des tâches réutilisables pour un lancement plus rapide sur vos appareils.

Rubriques

- Utiliser des modèles AWS gérés pour déployer des opérations à distance courantes
- Créer un modèle de tâche personnalisé

Utiliser des modèles AWS gérés pour déployer des opérations à distance courantes

AWS les modèles gérés sont des modèles de tâches fournis par AWS. Ils sont utilisés pour les actions à distance fréquemment effectuées, telles que le redémarrage, le téléchargement d'un fichier ou l'installation d'une application sur vos appareils. Ces modèles comportent un document de travail prédéfini pour chaque action à distance, de sorte que vous n'avez pas à créer votre propre document de travail.

Vous pouvez choisir parmi un ensemble de configurations prédéfinies et créer des tâches à l'aide de ces modèles sans écrire de code supplémentaire. À l'aide de modèles gérés, vous pouvez consulter le document de tâche déployé dans vos flottes. Vous pouvez créer une tâche à l'aide de ces modèles et créer un modèle de tâche personnalisé que vous pouvez réutiliser pour vos opérations à distance.

Que contiennent les modèles gérés ?

Chaque modèle AWS géré contient :

- L'environnement dans lequel exécuter les commandes du document de travail.
- Document de travail qui indique le nom de l'opération et ses paramètres. Par exemple, si vous utilisez un modèle de fichier de téléchargement, le nom de l'opération est Télécharger le fichier et les paramètres peuvent être les suivants :
  - Le URL fichier que vous souhaitez télécharger sur votre appareil. Il peut s'agir d'une ressource Internet ou d'un Amazon Simple Storage Service (Amazon S3) URL public ou pré-signé.
  - Un chemin de fichier local sur l'appareil pour stocker le fichier téléchargé.

  Pour plus d'informations sur la fonction et ses paramètres, veuillez consulter Modèles gérés d'actions à distance et de documents de tâche.

Prérequis

Pour que vos appareils exécutent les actions à distance spécifiées par le modèle de document de travail géré, vous devez :

- Installez le logiciel spécifique sur votre appareil

  Utilisez le logiciel et les gestionnaires de tâches de votre appareil, ou le client de l' AWS IoT appareil. En fonction de votre analyse de rentabilisation, vous pouvez également les exécuter tous les deux de manière à ce qu'ils remplissent des fonctions différentes.

  - Utilisez votre propre appareil, votre propre logiciel et vos propres gestionnaires de tâches

    Vous pouvez écrire votre propre code pour les appareils en utilisant Kit SDK des appareils AWS IoT et sa bibliothèque de gestionnaires qui prennent en charge les opérations à distance. Pour déployer et exécuter des tâches, vérifiez que les bibliothèques d'agents d'appareil ont été correctement installées et qu'elles s'exécutent sur les appareils.

    Vous pouvez également choisir d'utiliser vos propres gestionnaires qui prennent en charge les opérations à distance. Pour plus d'informations, consultez la section Exemples de gestionnaires de tâches dans le GitHub référentiel AWS IoT Device Client.

  - Utiliser le client de AWS IoT l'appareil

    Vous pouvez également installer et exécuter le AWS IoT Device Client sur vos appareils, car il prend en charge par défaut l'utilisation de tous les modèles gérés directement depuis la console.

    Le Device Client est un logiciel open source écrit en C++ que vous pouvez compiler et installer sur vos appareils IoT intégrés basés sur Linux. Le Device Client possède un client de base et des fonctionnalités distinctes côté client. Le client de base établit la connectivité AWS IoT via MQTT le protocole et peut se connecter aux différentes fonctionnalités côté client.

    Pour effectuer des opérations à distance sur vos appareils, utilisez la fonctionnalité Tâches côté client du Device Client. Cette fonctionnalité contient un analyseur pour recevoir le document de tâche et des gestionnaires de tâches qui implémentent les actions à distance spécifiées dans le document de travail. Pour plus d'informations sur Device Client et ses fonctionnalités, veuillez consulter AWS IoT Device Client.

    Lorsqu'il est exécuté sur des appareils, le Device Client reçoit le document de tâche et dispose d'une implémentation spécifique à la plate-forme qu'il utilise pour exécuter des commandes dans le document. Pour plus d'informations sur la configuration de Device Client et l'utilisation de la fonctionnalité Tâches, veuillez consulter AWS IoT les didacticiels..

- Utiliser l'environnement compatible

  Pour chaque modèle géré, vous trouverez des informations sur l'environnement que vous pouvez utiliser pour exécuter les actions à distance. Nous vous recommandons d'utiliser le modèle avec

un environnement Linux pris en charge comme indiqué dans le modèle. Utilisez le client de AWS IoT périphérique pour exécuter les actions à distance du modèle géré, car il prend en charge les microprocesseurs courants et les environnements Linux, tels que Debian et Ubuntu.

Modèles gérés d’actions à distance et de documents de tâche

La section suivante répertorie les différents modèles AWS gérés pour les AWS IoT tâches et décrit les actions à distance qui peuvent être effectuées sur les appareils. La section suivante contient des informations sur le document de travail et une description des paramètres du document de tâche pour chaque action à distance. Le logiciel côté appareil utilise le nom du modèle et les paramètres pour effectuer l’action à distance.

AWS les modèles gérés acceptent les paramètres d'entrée pour lesquels vous spécifiez une valeur lors de la création d'une tâche à l'aide du modèle. Tous les modèles gérés ont en commun deux paramètres d’entrée facultatifs : `runAsUser` et `pathToHandler`. À l’exception du modèle `AWS-Reboot`, les modèles nécessitent des paramètres d’entrée supplémentaires pour lesquels vous devez spécifier une valeur lors de la création d’une tâche à l’aide du modèle. Les paramètres d’entrée requis varient en fonction du modèle que vous choisissez. Par exemple, si vous choisissez le `AWS-Download-File` modèle, vous devez spécifier une liste de packages à installer et un package URL à partir duquel télécharger des fichiers.

Spécifiez une valeur pour les paramètres d'entrée lorsque vous utilisez la AWS IoT console ou le AWS Command Line Interface (AWS CLI) pour créer une tâche utilisant un modèle géré. Lorsque vous utilisez leCLI, fournissez ces valeurs à l'aide de l'`document-parameters`objet. Pour plus d'informations, consultez documentParameters.

> **Note**
>
> Utilisez `document-parameters` uniquement lors de la création de tâches à partir de modèles gérés AWS . Ce paramètre ne peut pas être utilisé avec des modèles de tâches personnalisés ou pour créer des tâches à partir de ceux-ci.

Vous trouverez ci-dessous une description des paramètres d’entrée facultatifs courants. Vous verrez une description des autres paramètres d’entrée requis par chaque modèle géré dans la section suivante.

`runAsUser`

Ce paramètre indique s'il faut exécuter le gestionnaire de tâches en tant qu'autre utilisateur. S'il n'est pas spécifié lors de la création de la tâche, le gestionnaire de tâches est exécuté sous le même nom d'utilisateur que le Device Client. Lorsque vous exécutez le gestionnaire de tâches en tant qu'autre utilisateur, spécifiez une valeur de chaîne ne dépassant pas 256 caractères.

`pathToHandler`

Le chemin d'accès au gestionnaire de tâches exécuté sur l'appareil. S'il n'est pas spécifié lors de la création de la tâche, le Device Client utilise le répertoire de tâche actuel.

Vous trouverez ci-dessous les différentes actions à distance, leurs documents de tâche et les paramètres qu'elles acceptent. Tous ces modèles prennent en charge l'environnement Linux pour exécuter l'opération à distance sur l'appareil.

AWS–Download–File

Nom du modèle

`AWS-Download-File`

Description du modèle

Modèle géré fourni par AWS pour le téléchargement d'un fichier.

Paramètres d'entrée

Les paramètres requis de ce modèle sont les suivants. Vous pouvez également spécifier les paramètres facultatifs `runAsUser` et `pathToHandler`.

`downloadUrl`

Le URL à partir duquel télécharger le fichier. Il peut s'agir d'une ressource Internet, d'un objet d'Amazon S3 accessible au public ou d'un objet d'Amazon S3 accessible uniquement par votre appareil à l'aide d'un document présignéURL. Pour plus d'informations sur l'utilisation des autorisations présignées URLs et l'octroi d'autorisations, consultezPrésigné URLs.

`filePath`

Un chemin de fichier local qui indique l'emplacement sur l'appareil où stocker le fichier téléchargé.

Comportement de l’appareil

L’appareil télécharge le fichier depuis l’emplacement indiqué, vérifie que le téléchargement est terminé et le stocke localement.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d’accès au gestionnaire de tâches et au script shell, `download-file.sh`, que le gestionnaire de tâches doit exécuter pour télécharger le fichier. Il indique également les paramètres requis `downloadUrl` et`filePath`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Download-File",
        "type": "runHandler",
        "input": {
          "handler": "download-file.sh",
          "args": [
            "${aws:iot:parameter:downloadUrl}",
            "${aws:iot:parameter:filePath}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Installer l’application.

Nom du modèle

`AWS-Install-Application`

Description du modèle

Modèle géré fourni par AWS pour l'installation d'une ou de plusieurs applications.

Paramètres d'entrée

Ce modèle possède le paramètre obligatoire suivant, `packages`. Vous pouvez également spécifier les paramètres facultatifs `runAsUser` et `pathToHandler`.

`packages`

Liste d’une ou plusieurs applications à installer, séparées par des espaces.

Comportement de l’appareil

L’appareil installe les applications comme indiqué dans le document de travail.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d’accès au gestionnaire de tâches et au script shell `install-packages.sh`, que le gestionnaire de tâches doit exécuter pour télécharger le fichier. Il indique également le paramètre requis `packages`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Install-Application",
        "type": "runHandler",
        "input": {
          "handler": "install-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Redémarrer

Nom du modèle

`AWS-Reboot`

Description du modèle

Un modèle géré fourni par AWS pour le redémarrage de votre appareil.

Paramètres d'entrée

Ce modèle n’a aucun paramètre requis. Vous pouvez également spécifier les paramètres facultatifs `runAsUser` et `pathToHandler`.

Comportement de l’appareil

L’appareil redémarre correctement.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d’accès au gestionnaire de tâches et au script shell `reboot.sh`, que le gestionnaire de tâches doit exécuter pour redémarrer l’appareil.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Suppression de l’application

Nom du modèle

`AWS-Remove-Application`

Description du modèle

Modèle géré fourni par AWS pour la désinstallation d'une ou de plusieurs applications.

Paramètres d'entrée

Ce modèle possède le paramètre obligatoire suivant, `packages`. Vous pouvez également spécifier les paramètres facultatifs `runAsUser` et `pathToHandler`.

`packages`

Liste d’une ou plusieurs applications à désinstaller, séparées par des espaces.

Comportement de l’appareil

L’appareil désinstalle les applications comme indiqué dans le document de tâche.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d’accès au gestionnaire de tâches et au script shell `remove-packages.sh`, que le gestionnaire de tâches doit exécuter pour télécharger le fichier. Il indique également le paramètre requis `packages`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Remove-Application",
        "type": "runHandler",
        "input": {
          "handler": "remove-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Redémarrage de l’application

Nom du modèle

`AWS-Restart-Application`

Description du modèle

Modèle géré fourni par AWS pour arrêter et redémarrer un ou plusieurs services.

Paramètres d'entrée

Ce modèle possède le paramètre obligatoire suivant, `services`. Vous pouvez également spécifier les paramètres facultatifs `runAsUser` et `pathToHandler`.

Services

Liste des applications, séparées par des espaces, à redémarrer.

Comportement de l’appareil

Les applications spécifiées sont arrêtées puis redémarrées sur l’appareil.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d’accès au gestionnaire de tâches et au script shell `restart-services.sh`, que le gestionnaire de tâches doit exécuter pour redémarrer les services du système. Il indique également le paramètre requis `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Restart-Application",
        "type": "runHandler",
        "input": {
          "handler": "restart-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
```

```
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Démarrage de l’application

Nom du modèle

`AWS-Start-Application`

Description du modèle

Modèle géré fourni par AWS pour démarrer un ou plusieurs services.

Paramètres d'entrée

Ce modèle possède le paramètre obligatoire suivant, `services`. Vous pouvez également spécifier les paramètres facultatifs `runAsUser` et `pathToHandler`.

`services`

Liste, séparées par des espaces, d’une ou plusieurs applications à démarrer.

Comportement de l’appareil

Les applications spécifiées commencent à s’exécuter sur l’appareil.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d’accès au gestionnaire de tâches et au script shell `start-services.sh`, que le gestionnaire de tâches doit exécuter pour démarrer les services système. Il indique également le paramètre requis `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
```

```
        "name": "Start-Application",
        "type": "runHandler",
        "input": {
          "handler": "start-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS–Arrêt de l’application

Nom du modèle

`AWS-Stop-Application`

Description du modèle

Modèle géré fourni par AWS pour arrêter un ou plusieurs services.

Paramètres d'entrée

Ce modèle possède le paramètre obligatoire suivant, `services`. Vous pouvez également spécifier les paramètres facultatifs `runAsUser` et `pathToHandler`.

`services`

Liste des applications, séparées par des espaces, à arrêter.

Comportement de l’appareil

Les applications spécifiées cessent de s’exécuter sur l’appareil.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d’accès au gestionnaire de tâches et au script shell `stop-services.sh`, que le gestionnaire de tâches doit exécuter pour arrêter les services système. Il indique également le paramètre requis `services`.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Stop-Application",
        "type": "runHandler",
        "input": {
          "handler": "stop-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

AWS Exécution d’une commande

Nom du modèle

`AWS-Run-Command`

Description du modèle

Modèle géré fourni par AWS pour exécuter une commande shell.

Paramètres d'entrée

Ce modèle possède le paramètre obligatoire suivant, `command`. Vous pouvez également spécifier le paramètre facultatif `runAsUser`.

`command`

Chaîne de commande séparée par des virgules. Toute virgule contenue dans la commande elle-même doit être évitée.

Comportement de l’appareil

Le périphérique exécute la commande shell comme indiqué dans le document de travail.

Document de tâche

Ce qui suit montre le document de travail et sa dernière version. Le modèle indique le chemin d'accès à la commande de tâche et à la commande que vous avez fournie et que l'appareil exécutera.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Run-Command",
        "type": "runCommand",
        "input": {
          "command": "${aws:iot:parameter:command}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Rubriques

- Créez une tâche à partir de modèles AWS gérés à l'aide du AWS Management Console
- Créez une tâche à partir de modèles AWS gérés à l'aide du AWS CLI

Créez une tâche à partir de modèles AWS gérés à l'aide du AWS Management Console

Utilisez le AWS Management Console pour obtenir des informations sur les modèles AWS gérés et créer une tâche à l'aide de ces modèles. Vous pouvez ensuite enregistrer la tâche que vous créez en tant que modèle personnalisé.

Obtenir des détails sur les modèles gérés

Vous pouvez obtenir des informations sur les différents modèles gérés qui peuvent être utilisés à partir de la AWS IoT console.

1. Pour voir les modèles gérés disponibles, accédez au hub de modèles de tâches de la AWS IoT console et choisissez l'onglet Modèles gérés.
2. Pour afficher les détails, choisissez un modèle géré.

La page des détails contient les informations suivantes :

- Nom, description et Amazon Resource Name (ARN) du modèle géré.
- L'environnement dans lequel les opérations à distance peuvent être effectuées, tel que Linux.
- Le document de JSON travail qui indique le chemin d'accès au gestionnaire de tâches et les commandes à exécuter sur le périphérique. Par exemple, ce qui suit montre un exemple de document de travail pour le modèle AWS-Reboot. Le modèle indique le chemin d'accès au gestionnaire de tâches et au script shell `reboot.sh`, que le gestionnaire de tâches doit exécuter pour redémarrer l'appareil.

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

  Pour plus d'informations sur le document de tâche et ses paramètres pour les différentes actions à distance, consultez Modèles gérés d'actions à distance et de documents de tâche.
- La dernière version du document de tâche.

Création d'une tâche à l'aide de modèles gérés

Vous pouvez utiliser la console AWS de gestion pour choisir un modèle AWS géré à utiliser pour créer une tâche. Cette section vous montre comment le faire.

Vous pouvez également démarrer le flux de travail de création de tâches, puis choisir le modèle AWS géré que vous souhaitez utiliser lors de la création de la tâche. Pour de plus amples informations

sur le contournement, veuillez consulter Créez et gérez des tâches à l'aide de AWS Management Console..

1. Choisissez votre modèle AWS géré

   Accédez au hub de modèles de tâches de la AWS IoT console, choisissez l'onglet Modèles gérés, puis choisissez votre modèle.

2. Créez une tâche à l'aide de votre modèle géré

   1. Sur la page de détails de votre modèle, choisissez Créer une tâche.

      La console passe à l'étape Propriétés de tâche personnalisées du flux de tâche de création de tâche dans laquelle la configuration de votre modèle a été ajoutée.

   2. Entrez un nom de tâche alphanumérique unique, ainsi qu'une description et des balises facultatives, puis choisissez Suivant.

   3. Choisissez les objets ou les groupes d'objets comme cibles de tâche que vous souhaitez exécuter dans cette tâche.

   4. Dans la section Document de tâche, votre modèle s'affiche avec ses paramètres de configuration et ses paramètres d'entrée. Entrez des valeurs pour les paramètres d'entrée du modèle que vous avez choisi. Par exemple, si vous avez choisi le modèle AWS-Download-File :

      - Pour downloadUrl, entrez URL le fichier à télécharger, par exemple :`https://example.com/index.html`.
      - Pour filePath, entrez le chemin sur l'appareil pour stocker le fichier téléchargé, par exemple :*path/to/file*.

      Vous pouvez également éventuellement saisir des valeurs pour les paramètres `runAsUser` et `pathToHandler`. Pour plus d'informations sur les paramètres de sortie de chaque modèle, veuillez consulter Modèles gérés d'actions à distance et de documents de tâche.

   5. Sur la page de configuration de la tâche, choisissez le type de tâche : tâche continue ou tâche instantanée. Une tâche de capture instantanée est terminée lorsqu'elle est exécutée sur les appareils et les groupes cibles. Une tâche continue s'applique aux groupes d'objets et s'exécute sur tous les appareils que vous ajoutez à un groupe cible spécifique.

   6. Continuez à ajouter des configurations supplémentaires pour votre tâche, puis passez en revue et créez votre tâche. Pour plus d'informations sur les configurations supplémentaires, veuillez consulter :

- Configurations du déploiement, de la planification et de l'annulation des tâches
- Configurations du délai d'exécution des tâches et des nouvelles tentatives

Créez des modèles de tâches personnalisés à partir de modèles gérés

Vous pouvez utiliser un modèle AWS géré et une tâche personnalisée comme point de départ pour créer votre propre modèle de tâche personnalisé. Pour créer un modèle de tâche personnalisé, créez d'abord une tâche à partir de votre modèle AWS géré, comme décrit dans la section précédente.

Vous pouvez ensuite enregistrer la tâche personnalisée en tant que modèle pour créer votre propre modèle de tâche personnalisé. Pour enregistrer en tant que modèle, procédez comme suit :

1. Accédez au Job hub de la AWS IoT console et choisissez le job contenant votre modèle géré.
2. Choisissez Enregistrer en tant que modèle de tâche, puis créez votre modèle de tâche personnalisé. Pour plus d'informations sur la création de modèles de tâche personnalisée, veuillez consulter Création d'un modèle de tâche à partir d'une tâche existante.

Créez une tâche à partir de modèles AWS gérés à l'aide du AWS CLI

Utilisez le AWS CLI pour obtenir des informations sur les modèles AWS gérés et créer une tâche à l'aide de ces modèles. Vous pouvez ensuite enregistrer le travail en tant que modèle, puis créer votre propre modèle personnalisé.

Répertorier les modèles gérés

La `list-managed-job-templates` AWS CLI commande répertorie tous les modèles de tâches de votre Compte AWS.

```
aws iot list-managed-job-templates
```

Par défaut, l'exécution de cette commande permet d'afficher tous les modèles AWS gérés disponibles et leurs détails.

```
{
    "managedJobTemplates": [
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Reboot:1.0",
```

```
            "templateName": "AWS-Reboot",
            "description": "A managed job template for rebooting the device.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Remove-
Application:1.0",
            "templateName": "AWS-Remove-Application",
            "description": "A managed job template for uninstalling one or more
 applications.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Stop-Application:1.0",
            "templateName": "AWS-Stop-Application",
            "description": "A managed job template for stopping one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },

        ...

        {
            "templateArn": "arn:aws:iot:us-east-1::jobtemplate/AWS-Restart-
Application:1.0",
            "templateName": "AWS-Restart-Application",
            "description": "A managed job template for restarting one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        }
    ]
```

```
}
```

Pour de plus amples informations, veuillez consulter ListManagedJobTemplates.

Récupère des détails sur un modèle géré

La `describe-managed-job-template` AWS CLI commande obtient des informations sur un modèle de tâche spécifié. Spécifiez le nom du modèle de tâche et une version de modèle facultative. Si la version du modèle n'est pas spécifiée, la version par défaut prédéfinie est renvoyée. L'exemple suivant montre comment exécuter la commande pour obtenir des détails sur le modèle `AWS-Download-File`.

```
aws iot describe-managed-job-template \
    --template-name AWS-Download-File
```

La commande affiche les détails du modèleARN, son document de travail et le `documentParameters` paramètre, qui est une liste de paires clé-valeur de paramètres d'entrée du modèle. Pour plus d'informations sur les différents modèles et paramètres d'entrée, consultez Modèles gérés d'actions à distance et de documents de tâche.

> **Note**
>
> L'`documentParameters`objet renvoyé lorsque vous l'utilisez ne API doit être utilisé que lors de la création de tâches à partir de modèles AWS gérés. L'objet ne doit pas être utilisé pour des modèles de tâches personnalisés. Pour obtenir un exemple pratique illustrant la façon d'utiliser ce paramètre, consultez Création d'une tâche à l'aide de modèles gérés.

```
{
    "templateName": "AWS-Download-File",
    "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0",
    "description": "A managed job template for downloading a file.",
    "templateVersion": "1.0",
    "environments": [
        "LINUX"
    ],
    "documentParameters": [
        {
```

```
            "key": "downloadUrl",
            "description": "URL of file to download.",
            "regex": "(.*?)",
            "example": "http://www.example.com/index.html",
            "optional": false
        },
        {
            "key": "filePath",
            "description": "Path on the device where downloaded file is written.",
            "regex": "(.*?)",
            "example": "/path/to/file",
            "optional": false
        },
        {
            "key": "runAsUser",
            "description": "Execute handler as another user. If not specified, then
 handler is executed as the same user as device client.",
            "regex": "(.){0,256}",
            "example": "user1",
            "optional": true
        },
        {
            "key": "pathToHandler",
            "description": "Path to handler on the device. If not specified, then
 device client will use the current working directory.",
            "regex": "(.){0,4096}",
            "example": "/path/to/handler/script",
            "optional": true
        }
    ],
    "document": "{\"version\":\"1.0\",\"steps\":[{\"action\":{\"name
\":\"Download-File\",\"type\":\"runHandler\",\"input\":{\"handler\":
\"download-file.sh\",\"args\":[\"${aws:iot:parameter:downloadUrl}\",
\"${aws:iot:parameter:filePath}\"],\"path\":\"${aws:iot:parameter:pathToHandler}\"},
\"runAsUser\":\"${aws:iot:parameter:runAsUser}\"}}]}"
}
```

Pour de plus amples informations, veuillez consulter DescribeManagedJobTemplate.

Création d'une tâche à l'aide de modèles gérés

La `create-job` AWS CLI commande peut être utilisée pour créer une tâche à partir d'un modèle de tâche. Il cible un appareil nommé `thing0ne` et spécifie le nom de ressource Amazon (ARN) du

modèle géré à utiliser comme base pour la tâche. Vous pouvez annuler les configurations avancées, telles que les configurations de temporisation et d'annulation, en transmettant les paramètres associés à la commande `create-job`.

L'exemple montre comment créer une tâche qui utilise le modèle `AWS-Download-File`. Il montre également comment spécifier les paramètres d'entrée du modèle à l'aide du paramètre `document-parameters`.

> **Note**
>
> Utilisez l'`document-parameters`objet uniquement avec des modèles AWS gérés. Cet objet ne doit pas être utilisé avec des modèles de tâches personnalisés.

```
aws iot create-job \
   --targets arn:aws:iot:region:account-id:thing/thingOne \
   --job-id "new-managed-template-job" \
   --job-template-arn arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0 \
   --document-parameters downloadUrl=https://example.com/index.html,filePath=path/to/
file
```

où :

- *region*est le Région AWS.
- *account-id*est le Compte AWS numéro unique.
- *thingOne* est le nom de l'objet IoT auquel le poste est destiné.
- *AWS-Download-File:1.0*est le nom du modèle géré.
- `https://`*example.com/index*`.html`est celui URL à partir duquel télécharger le fichier.
- `https://`*pathto/file/index* est le chemin sur l'appareil pour stocker le fichier téléchargé.

Exécutez la commande suivante pour créer une tâche pour le modèle *AWS-Download-File*.

```
{
    "jobArn": "arn:aws:iot:region:account-id:job/new-managed-template-job",
    "jobId": "new-managed-template-job",
    "description": "A managed job template for downloading a file."
}
```

Création d'un modèle de travail personnalisé à partir de modèles gérés

1. Créez une tâche à l'aide d'un modèle géré comme indiqué dans la section précédente.
2. Créez un modèle de tâche personnalisé en utilisant celui ARN de la tâche que vous avez créée. Pour de plus amples informations, veuillez consulter Création d'un modèle de tâche à partir d'une tâche existante.

Créer un modèle de tâche personnalisé

Vous pouvez créer des modèles de tâches à l'aide de la console AWS CLI et de la AWS IoT console. Vous pouvez également créer des tâches à partir de modèles de tâches à l' AWS CLI aide de la AWS IoT console et des applications Web Fleet Hub pour la gestion des AWS IoT appareils. Pour plus d'informations sur l'utilisation des modèles de tâches dans les applications Fleet Hub, voir Utilisation des modèles de tâches dans Fleet Hub pour la gestion des AWS IoT appareils.

> **Note**
>
> Le nombre total de modèles de substitution dans un document de tâche doit être inférieur ou égal à dix.

Rubriques

- Créez des modèles de tâches personnalisés à l'aide du AWS Management Console
- Créez des modèles de tâches personnalisés à l'aide du AWS CLI

Créez des modèles de tâches personnalisés à l'aide du AWS Management Console

Cette rubrique explique comment créer, supprimer et afficher les détails relatifs aux modèles de tâches à l'aide de la AWS IoT console.

Créer un modèle de tâche personnalisé

Vous pouvez créer un modèle de tâche personnalisé original ou créer un modèle de tâche à partir d'une tâche existante. Vous pouvez également créer un modèle de tâche personnalisé à partir d'une tâche existante créée à l'aide d'un modèle AWS géré. Pour de plus amples informations, veuillez consulter Créez des modèles de tâches personnalisés à partir de modèles gérés.

Création d'un modèle de travail original

1. Commencez à créer votre modèle de tâche

   1. Accédez au hub de modèles de tâches de la AWS IoT console et choisissez l'onglet Modèles personnalisés.
   2. Choisissez Créer un modèle de tâche.

   > **Note**
   >
   > Vous pouvez également accéder à la page des modèles de tâches depuis la page des services associés sous Fleet Hub.

2. Spécifier les propriétés du modèle de tâche

   Sur la page Créer un modèle de tâche, entrez un identifiant alphanumérique pour le nom de votre tâche et une description alphanumérique pour fournir des informations supplémentaires sur le modèle.

   > **Note**
   >
   > Nous vous déconseillons d'utiliser des informations personnellement identifiables dans votre poste IDs ou dans vos descriptions de poste.

3. Fournir un document de travail

   Fournissez un fichier de JSON travail qui est soit stocké dans un compartiment S3, soit sous forme de document de travail en ligne spécifié dans le travail. Ce fichier de travail deviendra le document de tâche lorsque vous créerez un tâche à l'aide de ce modèle.

   Si le fichier de travail est stocké dans un compartiment S3, entrez dans le S3 URL ou choisissez Parcourir S3, puis accédez à votre document de travail et sélectionnez-le.

   > **Note**
   >
   > Vous ne pouvez sélectionner que les compartiments S3 de votre région .

4. Continuez à ajouter des configurations supplémentaires pour votre tâche, puis passez en revue et créez votre tâche. Pour plus d'informations sur les configurations supplémentaires et facultatives, consultez les liens suivants :

   - Configurations du déploiement, de la planification et de l'annulation des tâches
   - Configurations du délai d'exécution des tâches et des nouvelles tentatives

Création d'un modèle de tâche à partir d'une tâche existante

1. Choisissez votre tâche

   1. Accédez au Job hub de la AWS IoT console et choisissez le job que vous souhaitez utiliser comme base pour votre modèle de job.
   2. Choisissez Enregistrer en tant que modèle de tâche.

   > **Note**
   >
   > Vous pouvez éventuellement choisir un autre document de tâche ou modifier les configurations avancées de la tâche d'origine, puis choisir Créer un modèle de tâche. Votre nouveau modèle de tâche apparaît sur la page Modèles de tâches.

2. Spécifier les propriétés du modèle de tâche

   Sur la page Créer un modèle de tâche, entrez un identifiant alphanumérique pour le nom de votre tâche et une description alphanumérique pour fournir des informations supplémentaires sur le modèle.

   > **Note**
   >
   > Le document de tâche est le fichier de tâche que vous avez spécifié lors de la création du modèle. Si le document de tâche est spécifié dans la tâche plutôt que dans un emplacement S3, vous pouvez voir le document de tâche sur la page de détails de cette tâche.

3. Continuez à ajouter des configurations supplémentaires pour votre tâche, puis passez en revue et créez votre tâche. Pour plus d'informations sur les configurations supplémentaires, veuillez consulter :

- Configurations du déploiement, de la planification et de l'annulation des tâches
- Configurations du délai d'exécution des tâches et des nouvelles tentatives

Créer une tâche à partir d'un modèle de tâche personnalisé

Vous pouvez créer une tâche à partir d'un modèle de tâche personnalisé en accédant à la page de détails de votre modèle de tâche, comme décrit dans cette rubrique. Vous pouvez également créer une tâche ou choisir le modèle de tâche que vous souhaitez utiliser lors de l'exécution du flux de travail de création de tâche. Pour de plus amples informations, veuillez consulter Créez et gérez des tâches à l'aide de AWS Management Console..

Cette rubrique explique comment créer une tâche à partir de la page de détails d'un modèle de tâche personnalisé. Vous pouvez également créer une tâche à partir d'un modèle AWS géré. Pour de plus amples informations, veuillez consulter Création d'une tâche à l'aide de modèles gérés.

1. Choisissez votre modèle de tâche personnalisé

   Accédez au hub de modèles de tâches de la AWS IoT console et choisissez l'onglet Modèles personnalisés, puis choisissez votre modèle.
2. Créez une tâche à l'aide de votre modèle personnalisé

   Pour créer une tâche :

   1. Sur la page de détails de votre modèle, choisissez Créer une tâche.

      La console passe à l'étape Propriétés de tâche personnalisées du flux de tâche de création de tâche dans laquelle la configuration de votre modèle a été ajoutée.
   2. Entrez un nom de tâche alphanumérique unique, ainsi qu'une description et des balises facultatives, puis choisissez Suivant.
   3. Choisissez les objets ou les groupes d'objets comme cibles de tâche que vous souhaitez exécuter dans cette tâche.

      Dans la section Document de tâche, votre modèle s'affiche avec ses paramètres de configuration. Si vous souhaitez utiliser un autre document de travail, choisissez Parcourir, puis sélectionnez un compartiment et un document différents. Choisissez Suivant.
   4. Sur la page de configuration de la tâche, choisissez le type de tâche : tâche continue ou tâche instantanée. Une tâche de capture instantanée est terminée lorsqu'elle est exécutée

sur les appareils et les groupes cibles. Une tâche continue s'applique aux groupes d'objets et s'exécute sur tous les appareils que vous ajoutez à un groupe cible spécifique.

5. Continuez à ajouter des configurations supplémentaires pour votre tâche, puis passez en revue et créez votre tâche. Pour plus d'informations sur les configurations supplémentaires, veuillez consulter :
   - Configurations du déploiement, de la planification et de l'annulation des tâches
   - Configurations du délai d'exécution des tâches et des nouvelles tentatives

> **Note**
>
> Lorsqu'une tâche créée à partir d'un modèle de tâche met à jour les paramètres existants fournis par le modèle de tâche, ces paramètres mis à jour remplacent les paramètres existants fournis par le modèle de tâche pour cette tâche.

Vous pouvez également créer des tâches à partir de modèles de tâches à l'aide des applications Web Fleet Hub. Pour plus d'informations sur la création de tâches dans Fleet Hub, voir Utilisation de modèles de tâches dans Fleet Hub pour la gestion des AWS IoT appareils.

Supprimer un modèle de tâche

Pour supprimer un modèle de tâche, accédez d'abord au hub de modèles de tâches de la AWS IoT console et choisissez l'onglet Modèles personnalisés. Ensuite, choisissez le modèle de tâche à supprimer, puis Suivant.

> **Note**
>
> Une suppression est définitive et le modèle de tâche n'apparaît plus dans l'onglet Modèles personnalisés.

Créez des modèles de tâches personnalisés à l'aide du AWS CLI

Cette rubrique explique comment créer, supprimer et récupérer des informations sur les modèles de tâches à l'aide de AWS CLI.

Créer une tâche à partir de zéro

La AWS CLI commande suivante montre comment créer une tâche à l'aide d'un document de tâche (*job-document.json*) stocké dans un compartiment S3 et d'un rôle autorisé à télécharger des fichiers depuis Amazon S3 (*S3DownloadRole*).

```
aws iot create-job-template  \
      --job-template-id 010  \
      --description "My custom job template for updating the device firmware"
      --document-source https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json
 \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\":
 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\":
 1000, \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType
\": \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20},
 { \"action\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings
\": 200, \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

Le paramètre `timeout-config` facultatif spécifie la durée allouée à chaque appareil pour terminer l'exécution de la tâche. Le minuteur est démarré quand l'état de l'exécution de la tâche a la valeur `IN_PROGRESS`. Si l'état de l'exécution de la tâche n'est pas défini sur un autre état terminal avant l'expiration, il est défini avec la valeur `TIMED_OUT`.

Le minuteur en cours ne peut pas être mis à jour et s'applique à toutes les lancements de tâche pour la tâche. Chaque fois qu'un lancement de tâche reste dans l'`IN_PROGRESS`état pendant plus longtemps que cet intervalle, le lancement de tâche échoue et passe à l'`TIMED_OUT`état du terminal. AWS IoT publie également une MQTT notification.

Pour plus d'informations sur la création de configurations relatives aux déploiements et interruptions de tâche, consultez la section Configuration du déploiement et de l'interruption des tâches.

> **Note**
>
> Les documents de tâche qui sont spécifiés en tant que fichiers Amazon S3 sont extraits au moment de la création de la tâche. Si vous modifiez le contenu du fichier Amazon S3 que

vous avez utilisé comme source de votre document de tâche après avoir créé la tâche, ce qui est envoyé aux cibles de la tâche ne change pas.

Création d'un modèle de tâche à partir d'une tâche existante

La AWS CLI commande suivante crée un modèle de tâche en spécifiant le nom de ressource Amazon (ARN) d'une tâche existante. Le nouveau modèle de tâche utilise toutes les configurations spécifiées dans la tâche. Vous pouvez éventuellement modifier n'importe quelle configuration de la tâche existante en utilisant l'un des paramètres facultatifs.

```
aws iot create-job-template  \
      --job-arn arn:aws:iot:region:123456789012:job/job-name  \
      --timeout-config inProgressTimeoutInMinutes=100
```

Récupère des détails sur un modèle de tâche

La AWS CLI commande suivante permet d'obtenir des informations sur un modèle de tâche spécifié.

```
aws iot describe-job-template \
      --job-template-id template-id
```

La commande affiche la sortie suivante :

```
{
   "abortConfig": {
      "criteriaList": [
         {
            "action": "string",
            "failureType": "string",
            "minNumberOfExecutedThings": number,
            "thresholdPercentage": number
         }
      ]
   },
```

```
   "createdAt": number,
   "description": "string",
   "document": "string",
   "documentSource": "string",
   "jobExecutionsRolloutConfig": {
      "exponentialRate": {
         "baseRatePerMinute": number,
         "incrementFactor": number,
         "rateIncreaseCriteria": {
            "numberOfNotifiedThings": number,
            "numberOfSucceededThings": number
         }
      },
      "maximumPerMinute": number
   },
   "jobTemplateArn": "string",
   "jobTemplateId": "string",
   "presignedUrlConfig": {
      "expiresInSec": number,
      "roleArn": "string"
   },
   "timeoutConfig": {
      "inProgressTimeoutInMinutes": number
   }
}
```

Répertorier les modèles de tâche

La AWS CLI commande suivante répertorie tous les modèles de tâches de votre Compte AWS.

```
aws iot list-job-templates
```

La commande affiche la sortie suivante :

```
{
   "jobTemplates": [
      {
         "createdAt": number,
         "description": "string",
```

```
            "jobTemplateArn": "string",
            "jobTemplateId": "string"
        }
    ],
    "nextToken": "string"
}
```

Pour récupérer des pages de résultats supplémentaires, utilisez la valeur du champ `nextToken`.

Supprimer un modèle de tâche

La AWS CLI commande suivante supprime un modèle de tâche spécifié.

```
aws iot delete-job-template \
      --job-template-id template-id
```

La commande n'affiche aucune sortie.

Créer une tâche à partir d'un modèle de tâche personnalisé

La AWS CLI commande suivante crée une tâche à partir d'un modèle de tâche personnalisé. Il cible un appareil nommé `thing0ne` et spécifie le nom de ressource Amazon (ARN) du modèle de tâche à utiliser comme base pour la tâche. Vous pouvez annuler les configurations avancées, telles que les configurations de temporisation et d'annulation, en transmettant les paramètres associés à la commande `create-job`.

> **Warning**
>
> L'objet `document-parameters` doit être utilisé avec la commande `create-job` uniquement lors de la création de tâches à partir de modèles gérés AWS . Cet objet ne doit pas être utilisé avec des modèles de tâches personnalisés. Pour obtenir un exemple pratique illustrant la création de tâches à l'aide de ce paramètre, consultez Création d'une tâche à l'aide de modèles gérés.

```
aws iot create-job \
```

```
    --targets arn:aws:iot:region:123456789012:thing/thingOne  \
    --job-template-arn arn:aws:iot:region:123456789012:jobtemplate/template-id
```

Configuration de la tâche

Vous pouvez disposer des configurations supplémentaires suivantes pour chaque tâche que vous déployez sur les cibles spécifiées.

- Déploiement : définit le nombre d'appareils recevant le document de tâche chaque minute.
- Planification : planifie une tâche pour une date et une heure futures en plus d'utiliser des fenêtres de maintenance récurrentes.
- Annulation : annule une tâche lorsque certains appareils ne reçoivent pas la notification de tâche ou lorsque vos appareils signalent un échec de l'exécution de la tâche.
- Délai d'attente : si vos objectifs de tâche ne répondent pas dans un certain délai après le début de leur exécution, la tâche peut échouer.
- Réessayer : Réessaie l'exécution de la tâche si votre appareil signale un échec lors de la tentative d'exécution d'une tâche, ou si le délai d'exécution de la tâche expire.

En utilisant ces configurations, vous pouvez surveiller l'état d'exécution de vos tâches et éviter qu'une mauvaise mise à jour ne soit envoyée à l'ensemble de la flotte.

Rubriques

- Comment fonctionnent les configurations de tâches
- Configurations supplémentaires

Comment fonctionnent les configurations de tâches

Vous utilisez les configurations de déploiement et d'annulation lorsque vous déployez une tâche, ainsi que les configurations de délai d'expiration et de nouvelle tentative pour l'exécution de la tâche. Les sections suivantes présentent plus d'informations sur le fonctionnement de ces configurations.

Rubriques

- Configurations du déploiement, de la planification et de l'annulation des tâches
- Configurations du délai d'exécution des tâches et des nouvelles tentatives

Configurations du déploiement, de la planification et de l'annulation des tâches

Vous pouvez utiliser les configurations de déploiement, de planification et d'annulation des tâches pour définir le nombre d'appareils recevant le document de tâche, planifier le déploiement d'une tâche et déterminer les critères d'annulation d'une tâche.

Configuration du déploiement des tâches

Vous pouvez spécifier la vitesse à laquelle les cibles sont averties d'une exécution de tâche en attente. Vous pouvez également créer un déploiement échelonné pour gérer les mises à jour, les redémarrages et d'autres opérations. Pour définir la manière dont vos cibles sont notifiées, utilisez les fréquences de déploiement des tâches.

Fréquences de déploiement des tâches

Vous pouvez créer une configuration de déploiement en utilisant une fréquence de déploiement constante ou une fréquence de déploiement exponentielle. Pour spécifier le nombre maximum d'objectifs de tâche à informer par minute, utilisez une fréquence de déploiement constante.

AWS IoT les emplois peuvent être déployés en utilisant des taux de déploiement exponentiels lorsque divers critères et seuils sont atteints. Si le nombre de tâches ayant échoué correspond à un ensemble de critères que vous spécifiez, vous pouvez annuler le déploiement des tâches. Vous définissez les critères de fréquence de déploiement des tâches lorsque vous créez une tâche à l'aide de l'objet `JobExecutionsRolloutConfig`. Les critères d'annulation de tâche sont définis lors de la création de la tâche via l'objet `AbortConfig`.

L'exemple suivant présente le fonctionnement des fréquence de déploiement. Par exemple, un déploiement de tâche avec une fréquence de base de 50 par minute, un facteur d'incrément de 2 et un nombre d'appareils notifiés et réussis égal à 1 000, fonctionnerait comme suit : la tâche débutera à un rythme de 50 exécutions par minute et se poursuivra à ce rythme jusqu'à ce que 1 000 objets aient reçu des notifications d'exécution de tâches ou que 1 000 exécutions de tâches aient eu lieu avec succès.

Le tableau suivant illustre la façon dont le déploiement procéderait sur les quatre premiers incréments.

Fréquence de lancement par minute	50	100	200	400

Nombre d'appareils notifiés ou d'exécutions de tâches réussies pour satisfaire une augmentation de fréquence	1 000	2 000	3 000	4 000

> **Note**
>
> Si vous avez atteint votre limite maximale de 500 tâches simultanées (`isConcurrent = True`), toutes les tâches actives conserveront le statut de `IN-PROGRESS` et ne lanceront aucune nouvelle exécution de tâches tant que le nombre de tâches simultanées ne sera pas inférieur ou égal à 499 (`isConcurrent = False`). Cela s'applique aux tâches instantanées et continues.
> Si `isConcurrent = True`, la tâche déploie actuellement des exécutions de tâches sur tous les appareils de votre groupe cible. Si `isConcurrent = False` la tâche a terminé le déploiement de toutes les exécutions de tâches sur tous les appareils de votre groupe cible. Il mettra à jour son état une fois que tous les appareils de votre groupe cible auront atteint l'état terminal, ou un pourcentage seuil de votre groupe cible si vous avez sélectionné une configuration d'annulation de tâche. Le statut du niveau de tâche indique pour `isConcurrent = True` et `isConcurrent = False` sont les deux `IN_PROGRESS`. Pour plus d'informations sur les limites des tâches actives et simultanées, consultez Limites de tâches actives et simultanées.

Fréquence de déploiement des tâches pour les tâches continues utilisant des groupes d'objets dynamiques

Lorsque vous utilisez une tâche continue pour déployer des opérations à distance sur votre flotte, AWS IoT Jobs exécute les tâches pour les appareils de votre groupe cible. Pour les nouveaux appareils ajoutés au groupe d'objets dynamiques, ces exécutions de tâches continuent d'être déployées sur ces appareils même après la création de la tâche.

La configuration de déploiement permet de contrôler les fréquence de déploiement uniquement pour les appareils ajoutés au groupe jusqu'à la création de la tâche. Après la création d'une tâche, pour tous les nouveaux appareils, les exécutions de tâches sont créées quasiment en temps réel dès que les appareils rejoignent le groupe cible.

Configuration de la planification des tâches

Vous pouvez planifier une tâche continue ou instantanée jusqu'à un an à l'avance en utilisant une heure de début, une heure de fin et un comportement de fin prédéterminés indiquant ce qu'il adviendra de chaque exécution de tâche une fois l'heure de fin atteinte. En outre, vous pouvez créer un créneau de maintenance récurrente facultative avec une fréquence, une heure de début et une durée flexibles pour les tâches continues afin de déployer un document de tâche sur tous les appareils du groupe cible.

Configurations de planification des tâches

L'heure de début

L'heure de début d'une tâche planifiée correspond à la date et à l'heure futures auxquelles la tâche commencera à être déployée du document de tâche sur tous les appareils du groupe cible. L'heure de début d'une tâche planifiée s'applique aux tâches continues et aux tâches instantanées. Lorsqu'une tâche planifiée est initialement créée, elle conserve un état de statut de `SCHEDULED`. Lorsque vous arrivez au document `startTime` que vous avez sélectionné, il est mis à jour à `IN_PROGRESS` et commence le déploiement du document de tâche. Le délai `startTime` doit être inférieur ou égal à un an à compter de la date et de l'heure initiales auxquelles vous avez créé la tâche planifiée.

Pour plus d'informations sur la syntaxe à utiliser `startTime` lors de l'utilisation d'une commande d'API ou du AWS CLI, consultez Timestamp.

Pour une tâche avec une configuration de planification optionnelle qui a lieu pendant un créneau de maintenance récurrente dans un lieu respectant l'heure d'été (DST), l'heure changera d'une heure lors du passage de l'heure d'été à l'heure normale et de l'heure standard à l'heure d'été.

> **Note**
>
> Le fuseau horaire affiché dans le AWS Management Console est le fuseau horaire actuel de votre système. Toutefois, ces fuseaux horaires seront convertis en UTC dans le système.

L'heure de fin

L'heure de fin d'une tâche planifiée est la date et l'heure futures auxquelles la tâche arrêtera le déploiement du document de tâche sur tous les appareils restants du groupe cible. L'heure de fin d'une tâche planifiée s'applique aux tâches continues et aux tâches instantanées. Une fois qu'une

tâche planifiée arrive à l'état sélectionné `endTime` et que toutes les exécutions de tâches ont atteint un état terminal, elle met à jour son état de statut de `IN_PROGRESS` à `COMPLETED`. Le délai `endTime` doit être inférieur ou égal à deux ans à compter de la date et de l'heure initiales auxquelles vous avez créé la tâche planifiée. La durée minimale entre `startTime` et `endTime` est de 30 minutes. Des tentatives de nouvelle tentative d'exécution de la tâche auront lieu jusqu'à ce que la tâche atteigne le `endTime`, puis `endBehavior` dicteront la marche à suivre.

Pour plus d'informations sur la syntaxe à utiliser `endTime` lors de l'utilisation d'une commande d'API ou du AWS CLI, consultez Timestamp.

Pour une tâche avec une configuration de planification optionnelle qui a lieu pendant un créneau de maintenance récurrente dans un lieu respectant l'heure d'été (DST), l'heure changera d'une heure lors du passage de l'heure d'été à l'heure normale et de l'heure standard à l'heure d'été.

> **Note**
>
> Le fuseau horaire affiché dans le AWS Management Console est le fuseau horaire actuel de votre système. Toutefois, ces fuseaux horaires seront convertis en UTC dans le système.

Comportement final

Le comportement final d'une tâche planifiée détermine ce qu'il advient de la tâche et de toutes les exécutions de tâches inachevées lorsque la tâche atteint la valeur sélectionnée `endTime`.

La liste suivante répertorie les comportements finaux que vous pouvez sélectionner lors de la création de la tâche ou du modèle de tâche :

- `STOP_ROLLOUT`
  - `STOP_ROLLOUT` arrête le déploiement du document de tâche sur tous les appareils restants du groupe cible de la tâche. De plus, toutes les exécutions de tâches `QUEUED` et `IN_PROGRESS` se poursuivront jusqu'à ce qu'elles atteignent l'état terminal. Il s'agit du comportement final par défaut, sauf si vous sélectionnez `CANCEL` ou`FORCE_CANCEL`.
- `CANCEL`
  - `CANCEL` arrête le déploiement du document de tâche sur tous les appareils restants du groupe cible de la tâche. En outre, toutes les exécutions de tâches `QUEUED` seront annulées et toutes les exécutions de tâches `IN_PROGRESS` se poursuivront jusqu'à ce qu'elles atteignent un état terminal.

- `FORCE_CANCEL`
  - `FORCE_CANCEL` arrête le déploiement du document de tâche sur tous les appareils restants du groupe cible de la tâche. De plus, toutes les exécutions de tâches `QUEUED` et `IN_PROGRESS` seront annulées.

> **Note**
>
> Pour sélectionner un`endbehavior`, vous devez sélectionner un `endtime`

Durée maximale

La durée maximale d'un tâche planifié doit être inférieure ou égale à deux ans, quel que soit le `startTime` et `endTime`.

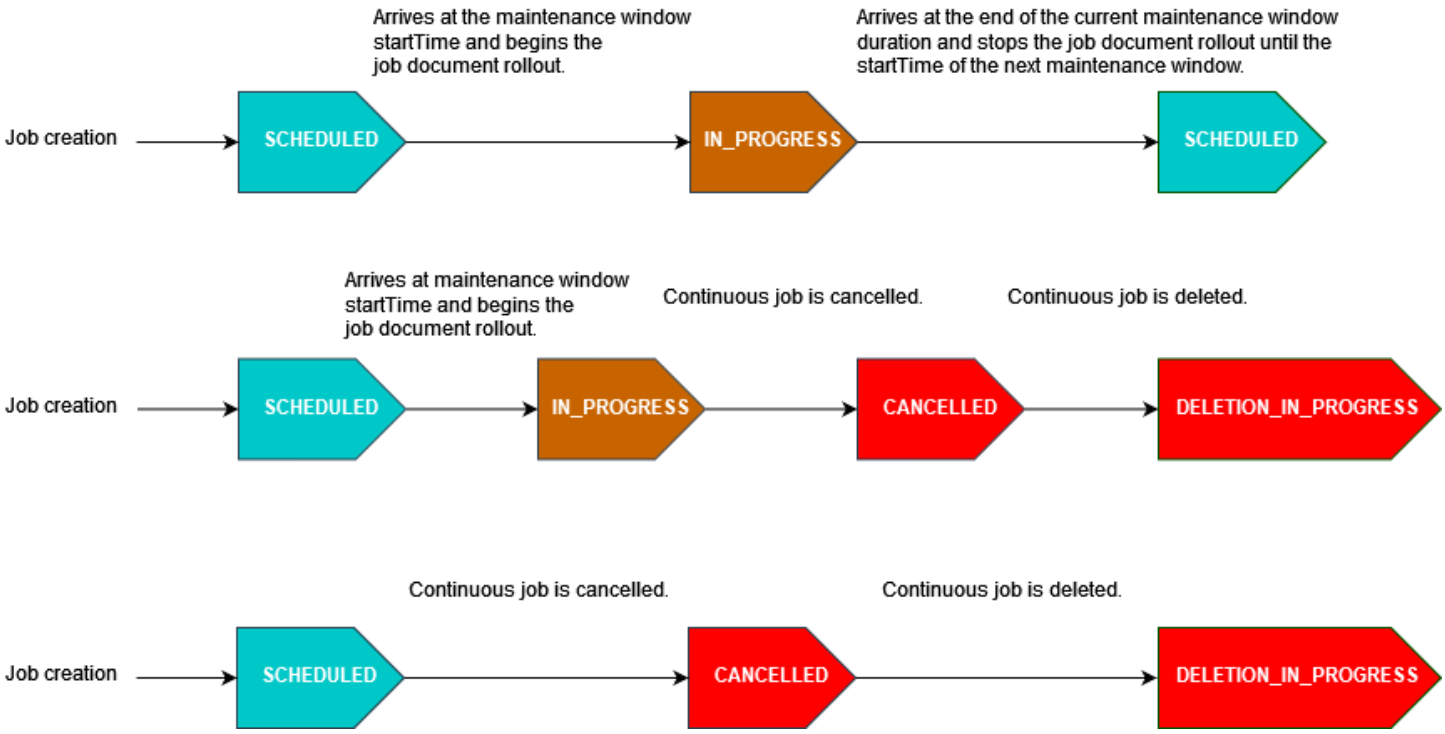
Le tableau suivant répertorie les scénarios de durée courants d'une tâche planifiée :

Numéro de l'exemple de tâche planifiée	startTime	endTime	Durée maximale
1	Immédiatement après la création initiale d'emplois.	Un an après la création initiale d'emplois.	Un an
2	Un mois après la création initiale de l'emploi.	13 mois après la création initiale de l'emploi.	Un an
3	Un an après la création initiale d'emplois.	Deux ans après la création initiale d'emplois.	Un an
4	Immédiatement après la création initiale d'emplois.	Deux ans après la création initiale d'emplois.	Deux ans

Créneau de maintenance récurrente

La fenêtre de maintenance est une configuration facultative dans la configuration de planification des `CreateJobTemplate` API AWS Management Console `CreateJob` et `SchedulingConfig` dans celles-ci. Vous pouvez configurer un créneau de maintenance récurrente avec une heure de début, une durée et une fréquence (quotidienne, hebdomadaire ou mensuelle) prédéterminées. Les créneaux de maintenance ne s'appliquent qu'aux tâches continues. La durée maximale d'un créneau de maintenance récurrente est de 23 heures et 50 minutes.

Le schéma suivant illustre l'état d'avancement des tâches pour différents scénarios de tâches planifiées avec un créneau de maintenance facultative :



Pour de plus amples informations sur l'état du statut d'une tâche, veuillez consulter Tâches et états d'exécution des tâches.

> **Note**
>
> Si une tâche arrive à `endTime` pendant une période de maintenance, elle sera mise à jour de `IN_PROGRESS` à `COMPLETED`. De plus, toutes les exécutions de tâches restantes suivront `endBehavior` pour la tâche.

Expressions Cron

Pour les tâches planifiées déployant le document de tâche pendant un créneau de maintenance avec une fréquence personnalisée, la fréquence personnalisée est saisie à l'aide d'une expression cron. Une expression cron se compose de six champs obligatoires qui sont séparés par des espaces.

Syntaxe

```
cron(fields)
```

Champ	Valeurs	Caractères génériques
Minutes	0-59	, - * /
Heures	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
Mois	1-12 ou JAN-DEC	, - * /
D ay-of-week	1-7 ou DIM-SAM	, - * ? L #
Année	1970-2199	, - * /

Caractères génériques

- Le caractère générique , (virgule) inclut des valeurs supplémentaires. Dans le champ Mois, JAN,FEB,MAR englobe janvier, février et mars.
- Le caractère générique - (tiret) spécifie des plages. Dans le champ Jour, 1-15 englobe les jours 1 à 15 du mois spécifié.
- Le caractère générique * (astérisque) inclut toutes les valeurs du champ. Dans le champ Hours, * inclut toutes les heures. Vous ne pouvez pas utiliser * à la fois dans les ay-of-week champs D ay-of-month et D. Si vous l'utilisez dans un champ, vous devez utiliser ? dans l'autre.
- Le caractère générique / (barre oblique) spécifie les incréments. Dans le champ Minutes, vous pouvez entrer 1/10 pour spécifier toutes les dix minutes, à partir de la première minute de l'heure (par exemple, les 11e, 21e, 31e minutes, et ainsi de suite).
- Le caractère générique ? (point d'interrogation) indique l'un ou l'autre. Dans le ay-of-month champ D, vous pouvez saisir 7 et si vous ne vous souciez pas du jour de la semaine le 7, vous pouvez entrer ? dans le ay-of-week champ D.

- Le caractère générique L dans les ay-of-week champs D ay-of-month ou D indique le dernier jour du mois ou de la semaine.
- Le **`W`** caractère générique dans le ay-of-month champ D indique un jour de la semaine. Dans le ay-of-month champ D, **`3W`** indique le jour de la semaine le plus proche du troisième jour du mois.
- Le caractère générique # dans le ay-of-week champ D indique une certaine instance du jour de la semaine spécifié dans un délai d'un mois. Par exemple, 3#2 correspond au deuxième mardi du mois : le 3 fait référence à mardi, car c'est le troisième jour de chaque semaine, et le 2 fait référence à la deuxième journée de ce type dans le mois.

> **Note**
>
> Si vous utilisez un caractère « # », vous ne pouvez définir qu'une seule expression dans le day-of-week champ. Par exemple, `"3#1,6#3"` n'est pas valide, car il est interprété comme deux expressions.

Restrictions

- Vous ne pouvez pas spécifier les ay-of-week champs D ay-of-month et D dans la même expression cron. Si vous spécifiez une valeur dans l'un de ces champs, vous devez utiliser un caractère générique ? dans l'autre.

Exemples

Reportez-vous aux exemples de chaînes cron suivants lorsque vous utilisez une expression cron pour la `startTime` de maintenance récurrente.

Minutes	Heures	Jour du mois	Mois	Jour de la semaine	Année	Signification
0 USD	10	*	*	?	*	Exécuter à 10 h 00 (UTC) chaque jour

Minutes	Heures	Jour du mois	Mois	Jour de la semaine	Année	Signification
15	12	*	*	?	*	Exécuter à 12 h 15 (UTC) chaque jour
0	18	?	*	MON-FRI	*	Exécuter à 18 h 00 (UTC) du lundi au vendredi
0	8	1	*	?	*	Exécuter à 8 h 00 (UTC) chaque 1er jour du mois

Durée du créneau de maintenance récurrente et logique de fin

Lorsqu'un déploiement de tâche pendant un créneau de maintenance atteint la fin de la durée d'occurrence du créneau de maintenance en cours, les actions suivantes se produisent :

- La tâche interrompra tous les déploiements du document de tâche sur tous les appareils restants de votre groupe cible. Il reprendra au créneau `startTime` de maintenance suivant.
- Toutes les exécutions de tâches dont le statut de `QUEUED` resteront dans `QUEUED` jusqu'à la `startTime` de la prochaine occurrence du créneau de maintenance. Dans le créneau suivant, ils peuvent passer à `IN_PROGRESS` au moment où l'appareil est prêt à commencer à exécuter les actions spécifiées dans le document de tâche.
- Toutes les exécutions de tâches dont le statut est `IN_PROGRESS` continueront d'exécuter les actions spécifiées dans le document de tâche jusqu'à ce qu'elles atteignent l'état terminal. Toute nouvelle tentative, comme indiqué dans `JobExecutionsRetryConfig` aura lieu au prochain créneau `startTime` de maintenance.

Configuration d’annulation de la tâche

Utilisez cette configuration pour créer un critère d’annulation d’une tâche lorsqu’un pourcentage seuil d’appareils répond à ces critères. Par exemple, vous pouvez utiliser cette configuration pour annuler une tâche dans les cas suivants :

- Lorsqu’un certain pourcentage d’appareils ne reçoivent pas les notifications d’exécution des tâches, par exemple lorsque votre appareil n’est pas compatible avec une mise à jour par voie hertzienne (OTA). Dans ce cas, votre appareil peut signaler un statut `REJECTED`.
- Lorsqu’un certain pourcentage d’appareils signalent l’échec de l’exécution de leurs tâches, par exemple lorsque votre appareil se déconnecte lorsqu’il tente de télécharger le document de tâche depuis une URL Amazon S3. Dans de tels cas, votre appareil doit être programmé pour signaler le statut `FAILURE` à AWS IoT.
- Lorsqu’un statut `TIMED_OUT` est signalé parce que l’exécution de la tâche expire pour un certain pourcentage d’appareils après le début de l’exécution de la tâche.
- En cas d’échec de plusieurs tentatives. Lorsque vous ajoutez une configuration de nouvelle tentative, chaque nouvelle tentative peut entraîner des frais supplémentaires pour votre Compte AWS. Dans de tels cas, l’annulation de la tâche peut annuler les exécutions de tâches en file d’attente et éviter de nouvelles tentatives pour ces exécutions. Pour plus d’informations sur la configuration de nouvelle tentative et son utilisation avec la configuration d’annulation, consultez Configurations du délai d’exécution des tâches et des nouvelles tentatives.

Vous pouvez configurer une condition d'abandon de tâche à l'aide de la AWS IoT console ou de l'API AWS IoT Jobs.

Configurations du délai d’exécution des tâches et des nouvelles tentatives

Utilisez la configuration du délai d’exécution des tâches pour vous envoyer Notifications Jobs lorsqu’une exécution de tâche est en cours depuis plus longtemps que la durée définie. Utilisez la configuration de nouvelle tentative d’exécution de la tâche pour réessayer l’exécution lorsque la tâche échoue ou expire.

Configuration du délai d’attente d’exécution de tâche

La configuration du délai d’exécution d’une tâche permet de vous avertir lorsqu’une tâche reste bloqué dans l’état `IN_PROGRESS` pendant une période de temps excessivement longue. Lorsque la tâche est `IN_PROGRESS`, vous pouvez suivre la progression de son exécution.

Minuteurs pour les délais d’expiration des tâches

Il existe deux types de minuteurs : minuteurs d'avancement et minuteurs d'étape.

Chronomètres en cours

Lorsque vous créez une tâche ou un modèle de tâche, vous pouvez spécifier une valeur comprise entre 1 minute et 7 jours pour le chronomètre en cours. Vous pouvez mettre à jour la valeur de ce chronomètre jusqu’au début de l’exécution de votre tâche. Une fois le chronomètre démarrée, il ne peut pas être mis à jour et la valeur du chronomètre s’applique à toutes les exécutions de tâches associées à la tâche. Chaque fois qu'une exécution de tâche reste dans le `IN_PROGRESS` statut pendant plus longtemps que cet intervalle, l'exécution de la tâche échoue et passe à l'`TIMED_OUT`état du terminal. AWS IoT publie également une notification MQTT.

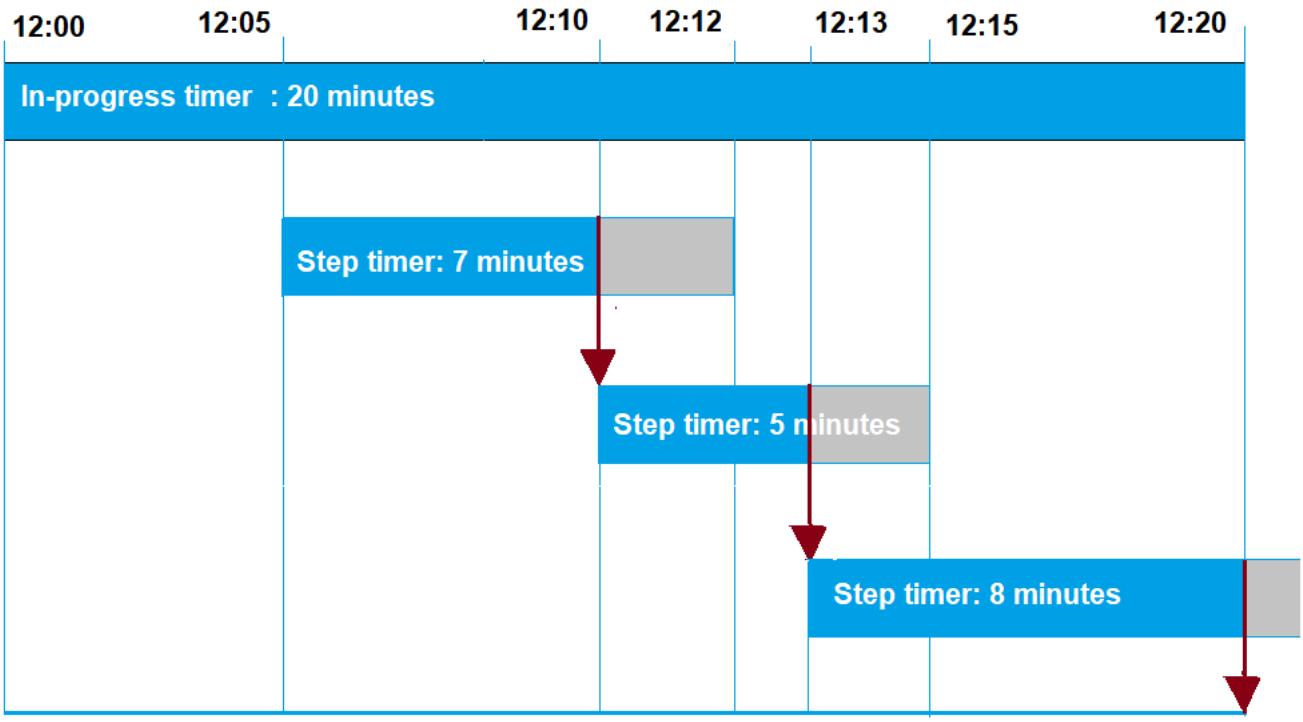
Chronomètre à étapes

Vous pouvez également définir un chronomètre qui s’applique uniquement à l’exécution de la tâche que vous souhaitez mettre à jour. Ce chronomètre n’a aucun effet sur le chronomètre en cours. Chaque fois que vous mettez à jour l’exécution d’une tâche, vous pouvez définir une nouvelle valeur pour la temporisation. Vous pouvez également créer un nouveau chronomètre lorsque vous démarrez l’exécution de tâche en attente suivante pour un objet. Si l'exécution de tâche demeure dans l'état `IN_PROGRESS` plus longtemps que l'intervalle du minuteur d'étape, elle échoue et passe à l'état final `TIMED_OUT`.

> **Note**
>
> Vous pouvez définir le chronomètre en cours à l'aide de la AWS IoT console ou de l'API AWS IoT Jobs. Pour spécifier le chronomètre, utilisez l’API.

Comment fonctionnent les chronomètres pour les délais d’expiration des tâches

Les exemples suivants illustrent les interactions entre les délais d’attente en cours et les délais d’attente par étape au cours d’une période de 20 minutes.



Les différentes étapes sont illustrées ci-dessous :

1. 12h00

   Une nouvelle tâche est créée et un chronomètre en cours de vingt minutes démarre lors de la création d'une tâche. Le chronomètre en cours commence à s'exécuter et l'exécution de la tâche passe au statut `IN_PROGRESS`.

2. 12h05

   Un nouveau chronomètre à étapes d'une valeur de 7 minutes est créé. L'exécution de la tâche expirera désormais à 12h12.

3. 12h10

   Un nouveau chronomètre à étapes d'une valeur de 5 minutes est créé. Lorsqu'un nouveau chronomètre est créé, le chronomètre à étapes précédent est supprimé et l'exécution de la tâche expire désormais à 12h15.

4. 12h13

   Un nouveau chronomètre à étapes d'une valeur de 9 minutes est créé. Le chronomètre à étapes précédent est supprimé et l'exécution de la tâche expirera désormais à 12h20 car le chronomètre

en cours expire à 12h20. Le chronomètre à étapes ne peut pas dépasser la limite absolue du chronomètre à étapes en cours.

Configuration de nouvelle tentative d'exécution de tâche

Vous pouvez utiliser la configuration de nouvelle tentative pour réessayer l'exécution de la tâche lorsqu'un certain ensemble de critères est satisfait. Une nouvelle tentative peut être tentée lorsqu'une tâche arrive à expiration ou lorsque l'appareil échoue. Pour réessayer l'exécution en raison d'un échec du délai d'attente, vous devez activer la configuration du délai d'expiration.

Comment utiliser la configuration de nouvelle tentative

Effectuez les étapes suivantes pour réessayer cette configuration.

1. Déterminez s'il convient d'utiliser la configuration de nouvelle tentative pour `FAILED`, `TIMED_OUT` ou les deux critères d'échec. En ce qui concerne le `TIMED_OUT` statut, une fois le statut signalé, AWS IoT Jobs réessaie automatiquement d'exécuter le travail pour l'appareil.
2. Pour connaître le statut `FAILED`, vérifiez si l'échec de l'exécution de votre tâche peut être réessayé. S'il est possible de réessayer, programmez votre appareil pour qu'il signale un statut `FAILURE` à AWS IoT. La section suivante décrit plus en détail les échecs réessayables et non réessayables.
3. Spécifiez le nombre de tentatives à utiliser pour chaque type d'échec à l'aide des informations précédentes. Pour un seul appareil, vous pouvez spécifier jusqu'à 10 tentatives pour les deux types d'échec combinés. Les tentatives de nouvelle tentative s'arrêtent automatiquement lorsqu'une exécution réussit ou lorsqu'elle atteint le nombre de tentatives spécifié.
4. Ajoutez une configuration d'annulation pour annuler le tâche en cas d'échecs répétés afin d'éviter des frais supplémentaires liés à un grand nombre de nouvelles tentatives.

> **Note**
>
> Lorsqu'une tâche arrive à la fin d'une période de maintenance récurrente, toutes les exécutions de tâches `IN_PROGRESS` continuent à exécuter les actions identifiées dans le document de tâche jusqu'à ce qu'elles atteignent l'état terminal. Si l'exécution d'une tâche atteint un état terminal de `FAILED` ou `TIMED_OUT` en dehors d'un créneau de maintenance, une nouvelle tentative aura lieu dans le créneau suivant si les tentatives ne sont pas épuisées. À la `startTime` du prochain créneau de maintenance, une nouvelle exécution

de tâche sera créée et entrera dans un état de `QUEUED` jusqu'à ce que l'appareil soit prêt à commencer.

Réessayer et annuler la configuration

Chaque nouvelle tentative entraîne des frais supplémentaires pour votre. Compte AWS Pour éviter d'encourir des frais supplémentaires en cas d'échecs répétés, nous vous recommandons d'ajouter une configuration d'annulation. Pour de plus amples informations sur la tarification, veuillez consulter AWS IoT Device Management Tarification.

Plusieurs tentatives peuvent échouer lorsqu'un pourcentage élevé de vos appareils expire ou signale un échec. Dans ce cas, vous pouvez utiliser la configuration d'annulation pour annuler la tâche et éviter toute exécution de tâche en file d'attente ou toute nouvelle tentative.

> **Note**
>
> Lorsque les critères d'annulation sont remplis pour annuler l'exécution d'une tâche, seules les exécutions de tâches `QUEUED` sont annulées. Aucune nouvelle tentative en file d'attente pour l'appareil ne sera tentée. Toutefois, les exécutions de tâches en cours qui ont un statut `IN_PROGRESS` ne seront pas annulées.

Avant de réessayer l'exécution d'une tâche qui a échoué, nous vous recommandons également de vérifier si l'échec de l'exécution de la tâche est réessayable, comme décrit dans la section suivante.

Réessayez en cas d'échec de type **`FAILED`**

Pour tenter une nouvelle tentative en cas d'échec de type `FAILED`, vos appareils doivent être programmés pour signaler l'état `FAILURE` de l'échec de l'exécution d'une tâche à AWS IoT. Définissez la configuration des nouvelles tentatives avec les critères permettant de réessayer l'exécution des tâches `FAILED` et spécifiez le nombre de tentatives à effectuer. Lorsque AWS IoT Jobs détecte l'`FAILURE`état, il tente automatiquement de réessayer d'exécuter le travail pour le périphérique. Les tentatives se poursuivent jusqu'à ce que l'exécution de la tâche réussisse ou jusqu'à ce que le nombre maximal de tentatives soit atteint.

Vous pouvez suivre chaque nouvelle tentative et le tâche en cours d'exécution sur ces appareils. En suivant l'état d'exécution, une fois que le nombre de tentatives spécifié a été atteint, vous pouvez utiliser votre appareil pour signaler les échecs et lancer une nouvelle tentative.

Défaillances réessayables et non réessayables

L'échec de l'exécution de votre tâche peut être réessayable ou non. Chaque nouvelle tentative peut entraîner des frais pour votre Compte AWS. Pour éviter d'encourir des frais supplémentaires en cas de tentatives multiples, pensez d'abord à vérifier si l'échec de l'exécution de votre tâche est réessayable. Un exemple d'échec réessayable inclut une erreur de connexion rencontrée par votre appareil lors de la tentative de téléchargement du document de tâche à partir d'une URL Amazon S3. Si l'échec de l'exécution de votre tâche est réessayable, programmez votre appareil pour qu'il signale un état `FAILURE` en cas d'échec de l'exécution de la tâche. Définissez ensuite la configuration de nouvelle tentative pour réessayer les exécutions `FAILED`.

Si l'exécution ne peut pas être relancée, afin d'éviter toute nouvelle tentative et d'entraîner des frais supplémentaires pour votre compte, nous vous recommandons de programmer l'appareil pour qu'il signale un statut `REJECTED` à AWS IoT. Parmi les échecs non réessayables, citons les cas où votre appareil n'est pas compatible avec la réception d'une mise à jour de tâche ou lorsqu'il rencontre une erreur de mémoire lors de l'exécution d'une tâche. Dans ces cas, AWS IoT Jobs ne réessaiera pas d'exécuter le travail car il ne recommencera l'exécution du travail que s'il détecte un statut `FAILED` ou`TIMED_OUT`.

Après avoir déterminé qu'un échec d'exécution d'une tâche est réessayable, si une nouvelle tentative échoue toujours, pensez à consulter les journaux de l'appareil.

> **Note**
>
> Lorsqu'une tâche avec la configuration de planification facultative atteint sa `endTime`, la commande `endBehavior` sélectionnée arrête le déploiement du document de tâche sur tous les appareils restants du groupe cible et dicte la marche à suivre pour les exécutions de tâches restantes. Les tentatives sont renouvelées si elles sont sélectionnées via la configuration de nouvelle tentative.

Réessayez en cas d'échec de type **`TIMEOUT`**

Si vous activez le délai d'expiration lors de la création d'une tâche, AWS IoT Jobs tentera de réessayer d'exécuter la tâche pour l'appareil lorsque le statut passe de à`IN_PROGRESS`. `TIMED_OUT` Ce changement d'état peut se produire lorsque le chronomètre en cours expire ou lorsqu'un chronomètre que vous spécifiez est `IN_PROGRESS` puis expire. Les tentatives se poursuivent jusqu'à ce que l'exécution de la tâche réussisse ou jusqu'à ce que le nombre maximal de tentatives soit atteint pour ce type d'échec.

Mises à jour continues sur les tâches et les membres des groupes d'objets

Pour les tâches continues dont le statut est égal à `IN_PROGRESS`, le nombre de nouvelles tentatives est remis à zéro lorsque l'appartenance au groupe d'un objet est mise à jour. Supposons, par exemple, que vous avez spécifié cinq tentatives et que trois tentatives ont déjà été effectuées. Si un objet est maintenant supprimé du groupe d'objets puis rejoint le groupe, par exemple dans le cas de groupes d'objets dynamiques, le nombre de nouvelles tentatives est remis à zéro. Vous pouvez désormais effectuer cinq tentatives pour votre groupe d'objets au lieu des deux tentatives restantes. En outre, lorsqu'un objet est supprimé du groupe d'objets, les tentatives supplémentaires sont annulées.

Configurations supplémentaires

Lorsque vous créez une tâche ou un modèle de tâche, vous pouvez spécifier ces configurations supplémentaires. Vous trouverez ci-dessous les cas dans lesquels vous pouvez définir ces configurations.

- Création d'un modèle de tâche personnalisé Les paramètres de configuration supplémentaires que vous spécifiez seront enregistrés lorsque vous créerez une tâche à partir du modèle.
- Lors de la création d'une tâche personnalisée à l'aide d'un fichier de tâche. Le fichier de tâche peut être un fichier JSON chargé dans un compartiment S3.
- Lorsque vous créez une tâche personnalisée à l'aide d'un modèle de tâche personnalisé. Si ces paramètres sont déjà spécifiés dans le modèle, vous pouvez les réutiliser ou les remplacer en spécifiant de nouveaux paramètres de configuration.
- Lors de la création d'une tâche personnalisée à l'aide d'un modèle AWS géré.

Rubriques

- Spécifiez les configurations des tâches à l'aide de la AWS Management Console
- Spécifiez les configurations des tâches à l'aide de l'API Jobs AWS IoT

Spécifiez les configurations des tâches à l'aide de la AWS Management Console

Vous pouvez ajouter les différentes configurations pour votre tâche à l'aide de la AWS IoT console. Une fois que vous avez créé une tâche, vous pouvez voir les détails du statut de vos configurations de tâche sur la page des détails de la tâche. Pour plus d'informations sur l'installation et la configuration des différents kit SDK, consultez Comment fonctionnent les configurations de tâches.

Ajoutez les configurations de tâche lorsque vous créez une tâche ou un modèle de tâche.

Création d'un modèle de tâche personnalisé

Pour spécifier la configuration du déploiement lors de la création d'un modèle de tâche personnalisé

1. Accédez au hub de modèles de tâches de la AWS IoT console et choisissez Create job template.
2. Spécifiez les propriétés du modèle de tâche, fournissez le document de tâche, développez la configuration que vous souhaitez ajouter, puis spécifiez les paramètres de configuration.

Lors de la création d'une tâche personnalisée

Pour spécifier la configuration du déploiement lors de la création d'une tâche personnalisée

1. Accédez au hub Job de la AWS IoT console et choisissez Create job.
2. Choisissez Créer une tâche personnalisée et spécifiez les propriétés de la tâche, les cibles et indiquez si vous souhaitez utiliser un fichier de tâche ou un modèle pour le document de tâche. Vous pouvez utiliser un modèle personnalisé ou un modèle AWS géré.
3. Choisissez la configuration de la tâche, puis développez la configuration de déploiement pour spécifier s'il faut utiliser une fréquence constante ou une fréquence exponentielle. Spécifiez ensuite les paramètres de configuration.

La section suivante présente les paramètres que vous pouvez spécifier pour chaque configuration.

Configuration du déploiement

Vous pouvez spécifier si vous souhaitez utiliser une fréquence de déploiement constant ou une fréquence exponentielle.

- Définissez une fréquence de déploiement constant

  Pour définir une fréquence constante pour les exécutions des tâches, choisissez fréquence constante, puis spécifiez le maximum par minute comme limite supérieure du fréquence. Cette valeur est facultative et est comprise entre 1 et 1 000. Si vous ne le définissez pas, il utilise 1000 comme valeur par défaut.
- Définissez une fréquence de déploiement exponentiel

  Pour définir une fréquence exponentielle, choisissez fréquence exponentielle, puis spécifiez les paramètres suivants :

- Tarif de base par minute

  La fréquence à laquelle les tâches sont exécutées jusqu'à ce que le seuil du nombre d'appareils notifiés ou du nombre d'appareils réussis soit atteint pour les critères d'augmentation du débit.
- Facteur d'incrément

  Le facteur exponentiel selon lequel le fréquence de déploiement augmente une fois que le seuil du nombre d'appareils notifiés ou du nombre d'appareils réussis est atteint pour les critères d'augmentation du fréquence.
- Critères d'augmentation de tarif

  Le seuil pour le nombre d'appareils notifiés ou le nombre d'appareils réussis.

Annulation de la configuration

Choisissez Ajouter une nouvelle configuration et spécifiez les paramètres suivants pour chaque configuration :

- Type de défaillance

  Spécifie les types d'échec qui déclenchent l'annulation d'une tâche. Il s'agit notamment de FAILED, REJECTED, TIMED_OUT ou ALL.
- Facteur d'incrément

  Spécifie le nombre d'exécutions de tâches terminées qui doivent se produire avant que les critères d'annulation des tâches ne soient remplies.
- Pourcentage de seuil

  Spécifie le nombre total de choses exécutées qui déclenchent l'annulation d'une tâche.

Configuration d'une planification.

Chaque tâche peut démarrer immédiatement après sa création initiale, être planifiée pour démarrer à une date et à une heure ultérieures, ou avoir lieu pendant un créneau de maintenance récurrente.

Choisissez Ajouter une nouvelle configuration et spécifiez les paramètres suivants pour chaque configuration :

- Démarrage d'une tâche

Spécifiez la date et l'heure de démarrage de la tâche ;

- Créneau de maintenance récurrente

  Un créneau de maintenance récurrente définit la date et l'heure spécifiques auxquelles une tâche peut déployer le document de tâche sur les équipements cibles de la tâche. Le créneau de maintenance peut être répété chaque jour, chaque semaine, chaque mois ou selon une périodicité personnalisée.

- Fin de la tâche

  Spécifiez la date et l'heure de la fin de tâche ;

- Comportement de fin de tâche

  Sélectionnez un comportement final pour toutes les exécutions de tâches inachevées une fois la tâche terminée.

> **Note**
>
> Lorsqu'une tâche avec la configuration de planification facultative et l'heure de fin sélectionnée atteint l'heure de fin, la tâche arrête le déploiement sur tous les appareils restants du groupe cible. Il tire également parti du comportement final sélectionné pour déterminer comment procéder aux exécutions de tâches restantes et à leurs tentatives de nouvelle tentative conformément à la configuration des nouvelles tentatives.

Configuration du délai d'attente

Par défaut, il n'y a pas de délai d'attente et votre tâche est annulée ou supprimée. Pour utiliser les délais d'attente, choisissez Activer le délai d'expiration, puis spécifiez une valeur de délai comprise entre 1 minute et 7 jours.

Nouvelle tentative de configuration

> **Note**
>
> Une fois qu'une tâche a été créée, le nombre de tentatives ne peut pas être mis à jour. Vous ne pouvez supprimer la configuration de nouvelle tentative que pour tous les types d'échec. Lorsque vous créez une tâche, considérez le nombre approprié de tentatives à utiliser pour

votre configuration. Pour éviter d'encourir des coûts supplémentaires en cas d'échec potentiel des nouvelles tentatives, ajoutez une configuration d'annulation.

Choisissez Ajouter une nouvelle configuration et spécifiez les paramètres suivants pour chaque configuration :

- Type de défaillance

  Spécifie les types d'échec qui doivent déclencher une nouvelle tentative d'exécution de la tâche. Il s'agit notamment de Échec, Expiré et Tout.
- Nombre de nouvelles tentatives

  Spécifie le nombre de tentatives pour le type d'échec choisi. Pour les deux types d'échec combinés, jusqu'à 10 tentatives peuvent être tentées.

Spécifiez les configurations des tâches à l'aide de l'API Jobs AWS IoT

Vous pouvez utiliser l'API CreateJobou l'CreateJobTemplateAPI pour spécifier les différentes configurations de travail. Les sections suivantes décrivent comment ajouter ces configurations. Après avoir ajouté les configurations, vous pouvez utiliser JobExecutionSummaryet JobExecutionSummaryForJobconsulter leur statut.

Pour plus d'informations sur l'installation et la configuration des différents kit SDK, consultez Comment fonctionnent les configurations de tâches.

Configuration du déploiement

Vous pouvez spécifier une fréquence de déploiement constante ou une fréquence de déploiement exponentielle pour votre configuration de déploiement.

- Définissez une fréquence de déploiement constante

  Pour définir une fréquence de déploiement constante, utilisez l'objet `JobExecutionsRolloutConfig` pour ajouter le paramètre `maximumPerMinute` à la demande `CreateJob`. Ce paramètre spécifie la limite supérieure de la fréquence à laquelle les exécutions de tâche peuvent se produire. Cette valeur est facultative et est comprise entre 1 et 1 000. Si vous ne définissez pas de valeur, elle utilise 1000 comme valeur par défaut.

```
    "jobExecutionsRolloutConfig": {
```

```
        "maximumPerMinute": 1000
    }
```

- Définissez une fréquence de déploiement exponentielle

  Pour définir une fréquence de déploiement des tâches variable, utilisez l'objet `JobExecutionsRolloutConfig`. Vous pouvez configurer la propriété `ExponentialRolloutRate` lorsque vous exécutez l'opération d'API `CreateJob`. L'exemple suivant définit une fréquence de déploiement exponentielle en utilisant le paramètre `exponentialRate`. Pour de plus amples informations sur les paramètres, veuillez consulter `ExponentialRolloutRate`.

```
{
...
  "jobExecutionsRolloutConfig": {
    "exponentialRate": {
      "baseRatePerMinute": 50,
      "incrementFactor": 2,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": 1000,
        "numberOfSucceededThings": 1000
      },
      "maximumPerMinute": 1000
    }
  }
...
}
```

Où le paramètre :

baseRatePerMinutes

Spécifie la fréquence à laquelle les tâches sont exécutées jusqu'à ce que le seuil `numberOfNotifiedThings` ou `numberOfSucceededThings` ait été atteint.

incrementFactor

Spécifie le facteur exponentiel par lequel la fréquence de déploiement augmente après que le seuil `numberOfNotifiedThings` ou `numberOfSucceededThings` a été atteint.

rateIncreaseCriteria

Spécifie le seuil `numberOfNotifiedThings` ou `numberOfSucceededThings`.

Annulation de la configuration

Pour ajouter cette configuration à l'aide de l'API, spécifiez le paramètre `AbortConfig` lorsque vous exécutez l'opération `CreateJob` ou l'opération de l'API `CreateJobTemplate`. L'exemple suivant montre une configuration d'annulation pour le déploiement d'une tâche qui a connu plusieurs échecs d'exécution, comme spécifié lors de l'opération d'API `CreateJob`.

> **Note**
>
> La suppression d'exécutions de tâche affecte la valeur de calcul de l'exécution totale achevée. Lorsqu'une tâche est annulée, le service crée un `comment` et un `reasonCode` automatiques pour différencier une annulation guidée par l'utilisateur d'une annulation par interruption de tâche.

```
"abortConfig": {
   "criteriaList": [
      {
         "action": "CANCEL",
         "failureType": "FAILED",
         "minNumberOfExecutedThings": 100,
         "thresholdPercentage": 20
      },
      {
         "action": "CANCEL",
         "failureType": "TIMED_OUT",
         "minNumberOfExecutedThings": 200,
         "thresholdPercentage": 50
      }
   ]
 }
```

Où le paramètre :

action

Spécifie l'action à entreprendre lorsque les critères d'annulation sont satisfaits. Ce paramètre est obligatoire et `CANCEL` est la seule valeur valide.

failureType

Spécifie les types d'échec qui doivent entraîner l'annulation d'une tâche. Les valeurs valides sont `FAILED`, `REJECTED`, `TIMED_OUT` et `ALL`.

minNumberOfExecutedThings

Le paramètre spécifie le nombre d'exécutions de tâches terminées qui doivent se produire avant que le service ne vérifie si les critères d'annulation de tâche ont été satisfaits. Dans cet exemple, AWS IoT ne vérifie pas si une annulation de tâche doit se produire tant que 100 appareils au moins n'ont pas terminé les exécutions de tâche.

thresholdPercentage

Spécifie le nombre total d'éléments pour lesquels les tâches sont exécutées et qui peuvent déclencher l'annulation d'une tâche. Dans cet exemple, AWS IoT effectue des vérifications séquentielles et lance un abandon de tâche si le pourcentage seuil est atteint. Si au moins 20 % des exécutions complètes échouent après 100 exécutions, le déploiement de la tâche est annulé. Si ce critère n'est pas rempli, AWS IoT vérifie si au moins 50 % des exécutions terminées ont expiré après la fin des 200 exécutions. Si tel est le cas, le déploiement de la tâche est annulé.

Configuration d'une planification.

Pour ajouter cette configuration à l'aide de l'API, spécifiez l'option `SchedulingConfig`lorsque vous exécutez l'opération `CreateJob` ou l'opération API `CreateJobTemplate`.

```
  "SchedulingConfig": {
    "endBehavior": string
    "endTime": string
    "maintenanceWindows": string
    "startTime": string
  }
```

Où le paramètre :

startTime

Spécifie la date et l'heure de lancement de la tâche.

endTime

Spécifie la date et l'heure de la fin de tâche.

maintenanceWindows

Spécifie si un créneau de maintenance facultative a été sélectionné pour la tâche planifiée afin de déployer le document de tâche sur tous les appareils du groupe cible. Le format de chaîne pour `maintenanceWindow` est AAAA/MM/JJ pour la date et hh:mm pour l'heure.

endBehavior

Spécifie le comportement d'une tâche planifiée lorsqu'elle atteint le `endTime`.

> **Note**
>
> Le `SchedulingConfig` facultatif pour une tâche est consultable dans le `DescribeJob` et les API `DescribeJobTemplate`.

Configuration du délai d'attente

Pour ajouter cette configuration à l'aide de l'API, spécifiez le paramètre `TimeoutConfig` lorsque vous exécutez l'opération `CreateJob` ou l'opération de l'API `CreateJobTemplate`.

Pour utiliser la configuration du délai d'expiration

1. Pour définir le chronomètre en cours lorsque vous créez une tâche ou un modèle de tâche, définissez une valeur pour la `inProgressTimeoutInMinutes` propriété de l'TimeoutConfigobjet facultatif.

```
 "timeoutConfig": {
   "inProgressTimeoutInMinutes": number
}
```

2. Pour spécifier un chronomètre pour l'exécution d'une tâche, définissez une valeur pour le `stepTimeoutInMinutes` moment où vous appelez UpdateJobExecution. Le minuteur d'étape s'applique uniquement à l'exécution des tâches que vous mettez à jour. Vous pouvez définir une nouvelle valeur pour ce minuteur chaque fois que vous mettez à jour une exécution de tâche.

> **Note**
>
> `UpdateJobExecution` peut également ignorer un chronomètre à étapes qui a déjà été créé en créant un chronomètre à étapes avec la valeur de -1.

```
{
   ...
    "statusDetails": {
      "string" : "string"
   },
   "stepTimeoutInMinutes": number
}
```

3. Pour créer un nouveau chronomètre, vous pouvez également appeler l'opération StartNextPendingJobExecutionAPI.

Nouvelle tentative de configuration

> **Note**
>
> Lorsque vous créez une tâche, considérez le nombre approprié de tentatives à utiliser pour votre configuration. Pour éviter d'encourir des coûts supplémentaires en cas d'échec potentiel des nouvelles tentatives, ajoutez une configuration d'annulation. Une fois qu'une tâche a été créée, le nombre de tentatives ne peut pas être mis à jour. Vous pouvez uniquement définir le nombre de tentatives à 0 à l'aide de l'opération UpdateJobAPI.

Pour ajouter cette configuration à l'aide de l'API, spécifiez le paramètre `jobExecutionsRetryConfig` lorsque vous exécutez l'opération `CreateJob` ou l'opération de l'API `CreateJobTemplate`.

```
{
...
  "jobExecutionsRetryConfig": {
      "criteriaList": [
         {
            "failureType": "string",
```

```
            "numberOfRetries": number
         }
      ]
  }
...
}
```

Où Criterialist est un tableau spécifiant la liste des critères qui déterminent le nombre de tentatives autorisées pour chaque type d'échec d'une tâche.

Appareils et tâches

Les appareils peuvent communiquer avec les AWS IoT tâches à l'aide de MQTT, de la signature HTTP version 4 ou du protocole HTTP TLS. Pour déterminer le point de terminaison à utiliser lorsque votre appareil communique avec AWS IoT Jobs, exécutez la DescribeEndpoint commande. Par exemple, si vous exécutez la commande suivante :

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

vous verrez une réponse comme celle-ci :

```
{
    "endpointAddress": "a1b2c3d4e5f6g7-ats.iot.us-west-2.amazonaws.com"
}
```

À l'aide du protocole MQTT

Les appareils peuvent communiquer avec les AWS IoT Jobs à l'aide du protocole MQTT. Les appareils s'abonnent aux rubriques MQTT pour être informés des nouvelles tâches et recevoir des réponses du service AWS IoT Jobs. Les appareils publient sur les rubriques MQTT pour interroger ou mettre à jour l'état de lancement d'une tâche. Chaque appareil a sa propre rubrique MQTT générale. Pour plus d'informations sur la publication et l'abonnement aux rubriques MQTT, consultez Protocoles de communication des appareils.

Avec cette méthode de communication, votre appareil utilise son certificat et sa clé privée spécifiques pour s'authentifier auprès de Jobs. AWS IoT

Vos appareils peuvent s'abonner aux rubriques suivantes. `thing-name` est le nom de l'élément associé à l'appareil.

- **$aws/things/*thing-name*/jobs/notify**

  Abonnez-vous à cette rubrique pour être averti lorsqu'un lancement de tâche est ajouté ou supprimé de la liste des lancements de tâches en attente.

- **$aws/things/*thing-name*/jobs/notify-next**

  Abonnez-vous à cette rubrique pour vous avertir lorsque l'exécution de la prochaine tâche en attente a changé.

- **$aws/things/*thing-name*/jobs/*request-name*/accepted**

  Le service AWS IoT Jobs publie des messages de réussite et d'échec sur un sujet MQTT. La rubrique est formée en ajoutant `accepted` ou `rejected` à la rubrique utilisée pour effectuer la demande. `request-name`Voici le nom d'une demande telle que `Get` et le sujet peut être :`$aws/things/myThing/jobs/get`. AWS IoT Jobs publie ensuite des messages de réussite sur le `$aws/things/myThing/jobs/get/accepted` sujet.

- **$aws/things/*thing-name*/jobs/*request-name*/rejected**

  `request-name`Voici le nom d'une demande telle que `Get`. Si la demande échoue, AWS IoT Jobs publie des messages d'échec sur le `$aws/things/myThing/jobs/get/rejected` sujet.

Vous pouvez également utiliser les opérations d'API HTTPS suivantes :

- Mettre à jour le statut d'une exécution de tâche en appelant l'API `UpdateJobExecution`.
- Interroger le statut d'une exécution de tâche en appelant l'API `DescribeJobExecution`.
- Récupérer la liste des exécutions de tâche en attente en appelant l'API `GetPendingJobExecutions`.
- Récupérer la prochaine exécution de tâche en attente en appelant l'API `DescribeJobExecution` avec `jobId` en tant que `$next`.
- Obtenir et démarrer la prochaine exécution de tâche en attente en appelant l'API `StartNextPendingJobExecution`.

À l'aide de HTTP Signature Version 4

Les appareils peuvent communiquer avec AWS IoT Jobs à l'aide de la signature HTTP version 4 sur le port 443. Il s'agit de la méthode utilisée par la CLI AWS SDKs et. Pour plus d'informations sur ces

outils, consultez la section Référence des AWS CLI commandes : iot-jobs-data ou AWS SDKs et Outils et reportez-vous à la IotJobsDataPlane section correspondant à votre langue préférée.

Avec cette méthode de communication, votre appareil utilise les informations d'identification IAM pour s'authentifier auprès de Jobs. AWS IoT

Les commandes suivantes sont disponibles à l'aide de cette méthode :

- DescribeJobExecution

```
aws iot-jobs-data describe-job-execution ...
```

- GetPendingJobExecutions

```
aws iot-jobs-data get-pending-job-executions ...
```

- StartNextPendingJobExecution

```
aws iot-jobs-data start-next-pending-job-execution ...
```

- UpdateJobExecution

```
aws iot-jobs-data update-job-execution ...
```

À l’aide de HTTP TLS

Les appareils peuvent communiquer avec AWS IoT Jobs à l'aide du protocole HTTP TLS sur le port 8443 à l'aide d'un client logiciel tiers prenant en charge ce protocole.

Avec cette méthode, votre appareil utilise l'authentification basée sur le certificat X.509 (par exemple, à l'aide du certificat et de la clé privée qui lui sont propres.)

Les commandes suivantes sont disponibles à l'aide de cette méthode :

- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

Programmation des appareils pour une utilisation avec Jobs

Les exemples de cette section utilisent MQTT pour illustrer la façon dont un appareil utilise le service AWS IoT Jobs. Vous pouvez également utiliser l'API ou les commandes CLI correspondantes. Pour ces exemples, nous supposons qu'un appareil appelé `MyThing` qui s'abonne aux rubriques MQTT suivantes :

- `$aws/things/MyThing/jobs/notify` (ou `$aws/things/MyThing/jobs/notify-next`)
- `$aws/things/MyThing/jobs/get/accepted`
- `$aws/things/MyThing/jobs/get/rejected`
- `$aws/things/MyThing/jobs/jobId/get/accepted`
- `$aws/things/MyThing/jobs/jobId/get/rejected`

Si vous utilisez la signature de code pour AWS IoT, le code de votre appareil doit vérifier la signature de votre fichier de code. La signature se trouve dans le document de tâche dans la propriété `codesign`. Pour plus d'informations sur la vérification d'une signature de fichier de code, consultez Exemple d'agent d'appareil.

Rubriques

- Flux de travail des appareils
- Flux de travail des tâches
- Notifications Jobs

Flux de travail des appareils

Un appareil peut gérer les tâches qu'il exécute de l'une des manières suivantes.

- Trouvez le prochain emploi

  1. Lorsqu'un appareil est mis en ligne pour la première fois, il doit s'abonner à la rubrique `notify-next` de l'appareil.
  2. Appelez l'API MQTT DescribeJobExecution avec le jobId `$next` pour obtenir la tâche suivante, son document de tâche et d'autres détails, y compris tout état enregistré dans `statusDetails`. Si le document de tâche possède une signature de fichier de code, vous devez vérifier la signature avant de continuer le traitement de la demande de tâche.

3. Appelez l'API MQTT UpdateJobExecution pour mettre à jour le statut de la tâche. Ou, pour combiner cette étape et la précédente en un seul appel, l'appareil peut appeler StartNextPendingJobExecution.
4. Le cas échéant, vous pouvez ajouter un minuteur d'étape en définissant une valeur pour `stepTimeoutInMinutes` lorsque vous appelez UpdateJobExecution ou StartNextPendingJobExecution.
5. Exécutez les actions spécifiées par le document de tâche à l'aide de l'API MQTT UpdateJobExecution pour faire état de l'avancement de la tâche.
6. Continuez de surveiller l'exécution de tâche en appelant l'API MQTT DescribeJobExecution avec ce jobId. Si l'exécution de la tâche est supprimée, DescribeJobExecution renvoie un `ResourceNotFoundException`.

   L'appareil doit être en mesure de revenir à un état valide si l'exécution de la tâche est annulée ou supprimée pendant que l'appareil exécute la tâche.
7. Une fois que la tâche est terminée, appelez l'API MQTT UpdateJobExecution pour mettre à jour le statut de la tâche et faire état de sa réussite ou de son échec.
8. Comme le statut d'exécution de cette tâche est passé à un état terminal, la prochaine tâche disponible pour une exécution (le cas échéant) change également. L'appareil est averti que la prochaine exécution de tâche en attente a changé. À ce stade, l'appareil doit continuer comme décrit à l'étape 2.

Si l'appareil reste en ligne, il continue de recevoir des notifications concernant l'exécution de la prochaine tâche en attente. Cela inclut ses données d'exécution de tâche, lorsqu'elle termine une tâche ou lorsqu'une nouvelle tâche en attente est ajoutée. Dans ce cas, l'appareil continue comme décrit à l'étape 2.

- Choisissez parmi les tâches disponibles

1. Lorsqu'un appareil est mis en ligne pour la première fois, il doit s'abonner à la rubrique `notify` de l'objet.
2. Appelez l'API MQTT GetPendingJobExecutions pour obtenir la liste des exécutions de tâche en attente.
3. Si la liste contient une ou plusieurs exécutions de tâche, sélectionnez-en une.
4. Appelez l'API MQTT DescribeJobExecution pour obtenir le document de tâche et autres détails, y compris tout état enregistré dans `statusDetails`.

5. Appelez l'API MQTT UpdateJobExecution pour mettre à jour le statut de la tâche. Si, dans cette commande, le champ `includeJobDocument` a la valeur `true`, l'appareil peut ignorer l'étape précédente et récupérer le document de tâche à ce stade.
6. Le cas échéant, vous pouvez ajouter un minuteur d'étape en définissant une valeur pour `stepTimeoutInMinutes` lorsque vous appelez UpdateJobExecution.
7. Exécutez les actions spécifiées par le document de tâche à l'aide de l'API MQTT UpdateJobExecution pour faire état de l'avancement de la tâche.
8. Continuez de surveiller l'exécution de tâche en appelant l'API MQTT DescribeJobExecution avec ce jobId. Si l'exécution de la tâche est annulée ou supprimée alors que l'appareil est en cours d'exécution, ce dernier doit être en mesure de retrouver un état valide.
9. Une fois que la tâche est terminée, appelez l'API MQTT UpdateJobExecution pour mettre à jour le statut de la tâche et faire état de sa réussite ou de son échec.

Si l'appareil demeure en ligne, il est averti de toutes les exécutions de tâche en attente chaque fois qu'une nouvelle exécution de tâche en attente devient disponible. Dans ce cas, l'appareil continue comme décrit à l'étape 2.

Si l'appareil n'est pas en mesure d'exécuter la tâche, il doit appeler l'API MQTT UpdateJobExecution pour mettre à jour le statut de la tâche avec la valeur `REJECTED`.

Flux de travail des tâches

Vous trouverez ci-dessous les différentes étapes du flux de travail des tâches, du démarrage d'une nouvelle tâche au signalement de l'état d'achèvement de l'exécution d'une tâche.

Démarrer une nouvelle tâche

Lorsqu'une nouvelle tâche est créée, AWS IoT Jobs publie un message sur le `$aws/things/`*`thing-name`*`/jobs/notify` sujet pour chaque appareil cible.

Le message contient les informations suivantes :

```
{
    "timestamp":1476214217017,
    "jobs":{
        "QUEUED":[{
            "jobId":"0001",
```

```
            "queuedAt":1476214216981,
            "lastUpdatedAt":1476214216981,
            "versionNumber" : 1
        }]
    }
}
```

L'appareil reçoit ce message sur la rubrique `'$aws/things/thingName/jobs/notify'` lorsque l'exécution de la tâche est en file d'attente.

> **Note**
>
> Pour les tâches dont `SchedulingConfig` est facultatif, la tâche conservera un statut initial de `SCHEDULED`. Lorsque la tâche atteint la valeur sélectionnée `startTime`, les événements suivants se produisent :
>
> - L'état du statut de la tâche sera mis à jour à `IN_PROGRESS`.
> - La tâche commencera à déployer le document de tâche sur tous les appareils du groupe cible.

Obtenir les informations sur une tâche

Pour obtenir plus d'informations sur l'exécution d'une tâche, l'appareil appelle l'API MQTT DescribeJobExecution avec le champ `includeJobDocument` défini sur `true` (valeur par défaut).

Si la demande aboutit, le service AWS IoT Jobs publie un message sur le `$aws/things/MyThing/jobs/0023/get/accepted` sujet :

```
{
    "clientToken" : "client-001",
    "timestamp" : 1489097434407,
    "execution" : {
        "approximateSecondsBeforeTimedOut": number,
        "jobId" : "023",
        "status" : "QUEUED",
        "queuedAt" : 1489097374841,
        "lastUpdatedAt" : 1489097374841,
        "versionNumber" : 1,
        "jobDocument" : {
            < contents of job document >
```

```
        }
    }
}
```

Si la demande échoue, le service AWS IoT Jobs publie un message sur le `$aws/things/MyThing/jobs/0023/get/rejected` sujet.

L'appareil dispose désormais du document de tâche, qu'il peut utiliser pour effectuer les opérations distantes pour la tâche. Si le document de tâche contient une URL Amazon S3 présignée, l'appareil peut utiliser cette URL pour télécharger les fichiers requis pour la tâche.

Rapport du statut d'exécution de tâche

Au fur et à mesure que l'appareil exécute la tâche, il peut appeler l'API MQTT UpdateJobExecution pour mettre à jour le statut de l'exécution de la tâche.

Par exemple, un appareil peut mettre à jour le statut de l'exécution de tâche `IN_PROGRESS` en publiant le message suivant sur la rubrique `$aws/things/MyThing/jobs/0023/update` :

```
{
    "status":"IN_PROGRESS",
    "statusDetails": {
        "progress":"50%"
    },
    "expectedVersion":"1",
    "clientToken":"client001"
}
```

Jobs répond en publiant un message dans la rubrique `$aws/things/MyThing/jobs/0023/update/accepted` ou `$aws/things/MyThing/jobs/0023/update/rejected` :

```
{
    "clientToken":"client001",
    "timestamp":1476289222841
}
```

Le périphérique permet de combiner les deux demandes précédentes en appelant StartNextPendingJobExecution. La prochaine exécution de tâche en attente est ainsi obtenue et démarrée et l'appareil peut mettre à jour le statut d'exécution de la tâche. La demande renvoie aussi le document de tâche lorsqu'il y a une exécution de tâche en attente.

Si la tâche contient un TimeoutConfig, le chronomètre en cours commence à fonctionner. Vous pouvez également définir un chronomètre pour l'exécution d'une tâche en définissant une valeur pour le `stepTimeoutInMinutes` moment où vous appelez UpdateJobExecution. Le minuteur d'étape s'applique uniquement à l'exécution des tâches que vous mettez à jour. Vous pouvez définir une nouvelle valeur pour ce minuteur chaque fois que vous mettez à jour une exécution de tâche. Vous pouvez également créer un chronomètre lorsque vous appelez StartNextPendingJobExecution. Si l'exécution de tâche demeure dans l'état `IN_PROGRESS` plus longtemps que l'intervalle du minuteur d'étape, elle échoue et passe à l'état final `TIMED_OUT`. Le minuteur d'étape n'a aucun effet sur le minuteur d'avancement que vous définissez lorsque vous créez une tâche.

Le champ `status` peut avoir la valeur `IN_PROGRESS`, `SUCCEEDED` ou `FAILED`. Vous ne pouvez pas mettre à jour le statut d'une exécution de tâche qui est déjà dans un état terminal.

Rapport d’exécution terminée

Lorsque l'appareil a terminé l'exécution de la tâche, il appelle l'API MQTT UpdateJobExecution. Si la tâche aboutit, définissez `status` sur `SUCCEEDED` et, dans le champ `statusDetails` de la charge utile du message, ajoutez d'autres informations sur la tâche sous la forme de paires nom-valeur. Les minuteurs d'avancement et d'étape prennent fin lorsque l'exécution de la tâche est terminée.

Par exemple :

```
{
    "status":"SUCCEEDED",
    "statusDetails": {
        "progress":"100%"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

Si la tâche n'a pas réussi, définissez `status` sur `FAILED` et, dans `statusDetails`, ajoutez les informations sur l'erreur qui s'est produite :

```
{
    "status":"FAILED",
    "statusDetails": {
        "errorCode":"101",
        "errorMsg":"Unable to install update"
    },
```

```
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

> **Note**
>
> L'attribut `statusDetails` peut contenir n'importe quel nombre de paires nom-valeur.

Lorsque le service AWS IoT Jobs reçoit cette mise à jour, il publie un message sur le `$aws/things/MyThing/jobs/notify` sujet pour indiquer que l'exécution de la tâche est terminée :

```
{
    "timestamp":1476290692776,
    "jobs":{}
}
```

Tâches supplémentaires

S'il existe d'autres exécutions de tâche en attente pour le périphérique, elles sont incluses dans le message publié sur `$aws/things/MyThing/jobs/notify`.

Par exemple :

```
{
    "timestamp":1476290692776,
    "jobs":{
        "QUEUED":[{
            "jobId":"0002",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }],
        "IN_PROGRESS":[{
            "jobId":"0003",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }]
    }
}
```

Notifications Jobs

Le service AWS IoT Jobs publie des messages MQTT sur des sujets réservés lorsque des tâches sont en attente ou lorsque la première exécution de tâche de la liste change. Les appareils peuvent suivre les tâches en attente en s'abonnant à ces rubriques.

Type de notification de tâche.

Les notifications de tâche sont publiées dans les rubriques MQTT en tant que charges utilises JSON. Il existe deux types de notifications :

ListNotification

Une `ListNotification` contient la liste de 15 exécutions de tâche en attente au plus. Elles sont triées par statut (les exécutions de tâche `IN_PROGRESS` précèdent les exécutions de tâche `QUEUED`), puis selon le moment auquel elles ont été mises en file d'attente.

Une `ListNotification` est publiée chaque fois que l'une des conditions ci-dessous est remplie.

- Une nouvelle exécution de tâche est mise en file d'attente ou passe à un statut qui n'est pas final (`IN_PROGRESS` ou `QUEUED`).
- Une ancienne exécution d'état acquiert le statut final (`FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` ou `REMOVED`).

Pour plus d'informations sur les limites avec et sans configuration de planification, consultezLimites relatives à l'exécution des tâches.

NextNotification

- Une `NextNotification` contient le résumé d'informations de l'exécution de tâche suivante dans la file d'attente.

  Une `NextNotification` est publiée chaque fois que la première exécution de tâche de la liste change.
  - Une nouvelle exécution de tâche est ajoutée à la liste au statut `QUEUED` et constitue le premier élément de la liste.
  - Le statut d'une exécution de tâche existante qui n'était pas le premier élément de la liste passe de `QUEUED` à `IN_PROGRESS` et devient le premier élément de la liste. (Cette situation se produit lorsque la liste ne contient aucune autre exécution de tâche `IN_PROGRESS` ou que l'exécution de

tâche dont le statut passe de `QUEUED` à `IN_PROGRESS` a été mise en file d'attente avant toutes les exécutions de tâche `IN_PROGRESS` de la liste.)

- Le statut de l'exécution de tâche qui est la première de la liste passe au statut final et est supprimée de la liste.

Pour plus d'informations sur la publication et l'abonnement aux rubriques MQTT, consultez the section called “Protocoles de communication des appareils”.

> **Note**
>
> Les notifications ne sont pas disponibles lorsque vous utilisez HTTP Signature Version 4 ou HTTP TLS pour communiquer avec les tâches.

Tâche en attente

Le service AWS IoT Jobs publie un message sur un sujet MQTT lorsqu'une tâche est ajoutée ou supprimée de la liste des tâches en attente pour un objet ou lorsque la première exécution de tâche de la liste change :

- `$aws/things/`*`thingName`*`/jobs/notify`
- `$aws/things/`*`thingName`*`/jobs/notify-next`

Les messages contiennent les exemples de charge utile suivants :

`$aws/things/`*`thingName`*`/jobs/notify`:

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : [ {
      "jobId" : "this-job",
```

```
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0
    } ]
  }
}
```

Si l'exécution de la tâche appelée `this-job` provient d'une tâche pour laquelle la configuration de planification facultative est sélectionnée et que le déploiement du document de tâche est prévu pour avoir lieu pendant une fenêtre de maintenance, il n'apparaîtra que pendant une fenêtre de maintenance récurrente. En dehors d'une fenêtre de maintenance, la tâche appelée `this-job` sera exclue de la liste des exécutions de tâches en attente, comme indiqué dans l'exemple suivant.

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : []
  }
}
```

`$aws/things/`*`thingName`*`/jobs/notify-next`:

```
{
  "timestamp" : 10011,
  "execution" : {
    "jobId" : "other-job",
    "status" : "IN_PROGRESS",
    "queuedAt" : 10009,
    "lastUpdatedAt" : 10009,
    "versionNumber" : 1,
    "executionNumber" : 1,
    "jobDocument" : {"c":"d"}
  }
}
```

Si l'exécution de la tâche appelée `other-job` provient d'une tâche pour laquelle la configuration de planification facultative est sélectionnée et que le déploiement du document de tâche est prévu pour avoir lieu pendant une fenêtre de maintenance, il n'apparaîtra que pendant une fenêtre de maintenance récurrente. En dehors d'une fenêtre de maintenance, la tâche appelée `other-job` ne sera pas répertoriée comme la prochaine exécution de la tâche, comme indiqué dans l'exemple suivant.

```
{} //No other pending jobs
```

```
{
  "timestamp" : 10011,
  "execution" : {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0,
      "jobDocument" : {"a":"b"}
  }
} // "this-job" is pending next to "other-job"
```

Les valeurs possibles d'état d'exécution de tâche sont `QUEUED`, `IN_PROGRESS`, `FAILED`, `SUCCEEDED`, `CANCELED`, `TIMED_OUT`, `REJECTED` et `REMOVED`.

La série d'exemples suivante montre les notifications publiées pour chaque rubrique lorsque des exécutions de tâches sont créées et passent d'un état à un autre.

D'abord, une tâche appelée `job1` est créée. Cette notification est publiée dans la rubrique `jobs/notify` :

```
{
  "timestamp": 1517016948,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      }
```

```
    ]
  }
}
```

Cette notification est publiée dans la rubrique `jobs/notify-next` :

```
{
  "timestamp": 1517016948,
  "execution": {
    "jobId": "job1",
    "status": "QUEUED",
    "queuedAt": 1517016947,
    "lastUpdatedAt": 1517016947,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Lorsqu'une autre tâche (`job2`) est créée, cette notification est publiée dans la rubrique `jobs/notify` :

```
{
  "timestamp": 1517017192,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
```

```
  }
}
```

Aucune notification n'est pas publiée dans la rubrique `jobs/notify-next`, car la tâche suivante de la file d'attente (`job1`) n'a pas changé. Lorsque `job1` commence à s'exécuter, son statut devient `IN_PROGRESS`. Aucune notification n'est publiée, car la liste des tâches et la tâche suivante dans la file d'attente n'ont pas changé.

Lorsqu'une troisième tâche (`job3`) est ajoutée, cette notification est publiée dans la rubrique `jobs/notify` :

```
{
  "timestamp": 1517017906,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517017472,
        "startedAt": 1517017472,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ],
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

Aucune notification n'est publiée dans la rubrique `jobs/notify-next`, car la tâche suivante de la file d'attente est toujours `job1`.

Une fois la tâche `job1` terminée, son statut passe à `SUCCEEDED` et cette notification est publiée dans la rubrique `jobs/notify` :

```
{
  "timestamp": 1517186269,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

À ce stade, la tâche `job1` a été supprimée de la file d'attente et la prochaine tâche à s'exécuter est `job2`. Cette notification est publiée dans la rubrique `jobs/notify-next` :

```
{
  "timestamp": 1517186269,
  "execution": {
    "jobId": "job2",
    "status": "QUEUED",
    "queuedAt": 1517017191,
    "lastUpdatedAt": 1517017191,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
```

```
    }
  }
}
```

Si la tâche `job3` doit commencer à s'exécuter avant la tâche `job2` (ce qui n'est pas recommandé), le statut de la tâche `job3` peut être modifié pour devenir `IN_PROGRESS`. Dans cette éventualité, `job2` n'est plus la tâche suivante dans la file d'attente et cette notification est publiée dans la rubrique `jobs/notify-next` :

```
{
  "timestamp": 1517186779,
  "execution": {
    "jobId": "job3",
    "status": "IN_PROGRESS",
    "queuedAt": 1517017905,
    "startedAt": 1517186779,
    "lastUpdatedAt": 1517186779,
    "versionNumber": 2,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

Aucune notification n'est publiée dans la rubrique `jobs/notify`, car aucune tâche n'a été ajoutée ou supprimée.

Si l'appareil rejette la tâche `job2` et met à jour son statut sur `REJECTED`, cette notification est publiée dans la rubrique `jobs/notify` :

```
{
  "timestamp": 1517189392,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517186779,
        "startedAt": 1517186779,
        "executionNumber": 1,
        "versionNumber": 2
```

```
      }
    ]
  }
}
```

Si la tâche `job3` (toujours en cours) est supprimée de force, cette notification est publiée dans la rubrique `jobs/notify` :

```
{
  "timestamp": 1517189551,
  "jobs": {}
}
```

À ce stade, la file d'attente est vide. Cette notification est publiée dans la rubrique `jobs/notify-next` :

```
{
  "timestamp": 1517189551
}
```

AWS IoT opérations de l'API jobs

AWS IoT L'API Jobs peut être utilisée pour l'une des catégories suivantes :

- Tâches administratives telles que la gestion et le contrôle des emplois. Voici le plan de contrôle.
- Appareils effectuant ces tâches. Il s’agit du plan de données qui vous permet d’envoyer et de recevoir des données.

La gestion et le contrôle des tâches utilisent une API du protocole HTTPS. Les appareils peuvent utiliser une API MQTT ou une API du protocole HTTPS. L’API du plan de contrôle est conçue pour un faible volume d’appels lors de la création et du suivi des tâches. Elle ouvre généralement une connexion pour une seule demande, puis la ferme après réception de la réponse. Le plan de données HTTPS et l’API MQTT permettent de longues interrogations. Ces opérations d’API sont conçues pour d’importantes quantités de trafic qui peuvent mettre à l’échelle plusieurs millions d’appareils.)

Chaque API HTTPS AWS IoT Jobs possède une commande correspondante qui vous permet d'appeler l'API depuis le AWS Command Line Interface (AWS CLI). Les commandes sont en

minuscules, avec un trait d'union entre les mots qui composent le nom de l'API. Par exemple, vous pouvez appeler l'API `CreateJob` sur l'interface de ligne de commande en tapant :

```
aws iot create-job ...
```

Si une erreur se produit pendant une opération, vous obtenez une réponse d’erreur contenant des informations sur l’erreur.

ErrorResponse

Contient les informations sur une erreur qui s'est produite au cours d'une opération du service AWS IoT Jobs.

L’exemple suivant illustre la syntaxe de cette opération :

```
{
    "code": "ErrorCode",
    "message": "string",
    "clientToken": "string",
    "timestamp": timestamp,
    "executionState": JobExecutionState
}
```

Voici une description de ce `ErrorResponse` :

`code`

ErrorCode peut être réglé sur :

InvalidTopic

La demande a été envoyée à une rubrique de l'espace de noms AWS IoT Jobs qui ne correspond à aucune opération d'API.

InvalidJson

Le contenu de la demande n'a pas pu être interprété comme JSON codé en UTF-8 valide.

InvalidRequest

Le contenu de la demande n’était pas valide. Par exemple, ce code est renvoyé lorsqu'une demande `UpdateJobExecution` contient des détails d'état non valides. Le message contient des détails sur l'erreur.

InvalidStateTransition

Une mise à jour a tenté de faire passer l'exécution de la tâche à un état non valide en raison de l'état actuel de l'exécution de la tâche. Par exemple, une tentative de modification d'une demande dont l'état est SUCCEDED par le statut IN_PROGRESS. Dans ce cas, le corps du message d'erreur contient aussi le champ `executionState`.

ResourceNotFound

La valeur `JobExecution` spécifiée par la rubrique de la demande n'existe pas.

VersionMismatch

La version attendue spécifiée dans la demande ne correspond pas à la version de l'exécution de la tâche dans le service AWS IoT Jobs. Dans ce cas, le corps du message d'erreur contient aussi le champ `executionState`.

InternalError

Une erreur interne s'est produite pendant le traitement de la demande.

RequestThrottled

La demande a été limitée.

TerminalStateReached

Se produit quand une commande pour décrire une tâche est exécutée sur une tâche qui se trouve dans un état terminal.

`message`

Chaîne de message d'erreur.

`clientToken`

Chaîne arbitraire utilisée pour mettre en corrélation une demande et sa réponse.

`timestamp`

Nombre de secondes depuis la date epoch Unix.

`executionState`

Un objet `JobExecutionState`. Ce champ est inclus uniquement lorsque le champ `code` a la valeur `InvalidStateTransition` ou `VersionMismatch`. Il est donc inutile dans ces cas-là

d'effectuer une demande `DescribeJobExecution` distincte pour obtenir les données du statut d'exécution de tâche en cours.

La liste suivante répertorie les opérations et les types de données de l'API de tâches.

- Types de données de gestion et de contrôle des tâches d'API
- Opérations des API MQTT et HTTPS de l'appareil et types de données

Types de données de gestion et de contrôle des tâches d'API

Les commandes suivantes sont disponibles pour la gestion et le contrôle des tâches dans la CLI et via le protocole HTTPS.

- Types de données de gestion et de contrôle des tâches
- Opérations d'API de gestion et de contrôle des tâches

Pour déterminer le *`endpoint-url`* paramètre de vos commandes CLI, exécutez cette commande.

```
aws iot describe-endpoint --endpoint-type=iot:Jobs
```

Cette commande renvoie la sortie suivante.

```
{
"endpointAddress": "account-specific-prefix.jobs.iot.aws-region.amazonaws.com"
}
```

> **Note**
>
> Le point de terminaison de tâches ne prend pas en charge ALPN `x-amzn-http-ca`.

Types de données de gestion et de contrôle des tâches

Les types de données suivants sont utilisés par les applications de gestion et de contrôle pour communiquer avec AWS IoT Jobs.

Tâche

L'objet Job contient des informations sur une tâche. L'exemple suivant montre la syntaxe :

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED",
    "forceCanceled": boolean,
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "comment": "string",
    "targets": ["string"],
    "description": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp,
    "jobProcessDetails": {
        "processingTargets": ["string"],
        "numberOfCanceledThings": long,
        "numberOfSucceededThings": long,
        "numberOfFailedThings": long,
        "numberOfRejectedThings": long,
        "numberOfQueuedThings": long,
        "numberOfInProgressThings": long,
        "numberOfRemovedThings": long,
        "numberOfTimedOutThings": long
    },
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
           "baseRatePerMinute": integer,
           "incrementFactor": integer,
           "rateIncreaseCriteria": {
              "numberOfNotifiedThings": integer, // Set one or the other
              "numberOfSucceededThings": integer // of these two values.
           },
           "maximumPerMinute": integer
      }
    },
    "abortConfig": {
       "criteriaList": [
```

```
          {
             "action": "string",
             "failureType": "string",
             "minNumberOfExecutedThings": integer,
             "thresholdPercentage": integer
          }
       ]
    },
    "SchedulingConfig": {
      "startTime": string
      "endTime": string
      "timeZone": string


      "endTimeBehavior": string

   },
    "timeoutConfig": {
        "inProgressTimeoutInMinutes": long
    }
}
```

Pour plus d'informations, consultez Job ou job.

JobSummary

L'objet JobSummary contient un résumé de tâche. L'exemple suivant montre la syntaxe :

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED|SCHEDULED",
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "thingGroupId": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp
}
```

Pour plus d'informations, consultez JobSummary ou job-summary.

JobExecution

L'objet `JobExecution` représente l'exécution d'une tâche sur un appareil. L'exemple suivant montre la syntaxe :

> **Note**
>
> Lorsque vous utilisez les opérations de l'API du plan de contrôle, le type de données `JobExecution` ne contient aucun champ `JobDocument`. Pour obtenir ces informations, vous pouvez utiliser l'opération d'API `GetJobDocument` ou la commande CLI `get-job-document`.

```
{
    "approximateSecondsBeforeTimedOut": 50,
    "executionNumber": 1234567890,
    "forceCanceled": true|false,
    "jobId": "string",
    "lastUpdatedAt": timestamp,
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "forceCanceled": boolean,
    "statusDetails": {
        "detailsMap": {
            "string": "string" ...
        },
        "status": "string"
    },
    "thingArn": "string",
    "versionNumber": 123
}
```

Pour plus d'informations, consultez `JobExecution` ou `job-execution`.

JobExecutionSummary

L'objet `JobExecutionSummary` contient les informations récapitulatives sur l'exécution de tâche. L'exemple suivant montre la syntaxe :

```
{
```

```
    "executionNumber": 1234567890,
    "queuedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"
}
```

Pour plus d'informations, consultez JobExecutionSummary ou job-execution-summary.

JobExecutionSummaryForJob

L'objet JobExecutionSummaryForJob contient un récapitulatif des informations sur les exécutions de tâche d'une tâche spécifique. L'exemple suivant montre la syntaxe :

```
{
    "executionSummaries": [
        {
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            }
        },
        ...
    ]
}
```

Pour plus d'informations, consultez JobExecutionSummaryForJob ou job-execution-summary-for-job.

JobExecutionSummaryForThing

L'JobExecutionSummaryForThingobjet contient un résumé des informations relatives à l'exécution d'une tâche sur un objet spécifique. FThe l'exemple suivant montre la syntaxe :

```
{
    "executionSummaries": [
        {
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
```

```
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            },
            "jobId": "MyThingJob"
        },
        ...
    ]
}
```

Pour plus d'informations, consultez JobExecutionSummaryForThing ou job-execution-summary-for-thing.

Opérations d'API de gestion et de contrôle des tâches

Utilisez les opérations API ou les commandes CLI suivantes :

AssociateTargetsWithJob

Associe un groupe à une tâche continue. Les critères suivants doivent être satisfaits :

- Lors de la création de la tâche, le champ `targetSelection` doit être défini sur `CONTINUOUS`.
- Le statut de la tâche doit actuellement être `IN_PROGRESS`.
- Le nombre total de cibles associées à une tâche ne doit pas dépasser 100.

HTTPS request

```
POST /jobs/jobId/targets

{
"targets": [ "string" ],
"comment": "string"
}
```

Pour de plus amples informations, veuillez consulter AssociateTargetsWithJob.

CLI syntax

```
aws iot  associate-targets-with-job \
--targets <value> \
--job-id <value> \
[--comment <value>]  \
```

```
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"targets": [
"string"
],
"jobId": "string",
"comment": "string"
}
```

Pour de plus amples informations, veuillez consulter `associate-targets-with-job`.

CancelJob

Annule une tâche.

HTTPS request

```
PUT /jobs/jobId/cancel

{
"force": boolean,
"comment": "string",
"reasonCode": "string"
}
```

Pour de plus amples informations, veuillez consulter `CancelJob`.

CLI syntax

```
aws iot cancel-job \
    --job-id <value> \
    [--force <value>]  \
    [--comment <value>]  \
    [--reasonCode <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
    "jobId": "string",
    "force": boolean,
    "comment": "string"
}
```

Pour de plus amples informations, veuillez consulter `cancel-job`.

CancelJobExecution

Annule une exécution de tâche sur un appareil.

HTTPS request

```
PUT /things/thingName/jobs/jobId/cancel

{
"force": boolean,
"expectedVersion": "string",
"statusDetails": {
    "string": "string"
    ...
}
}
```

Pour de plus amples informations, veuillez consulter `CancelJobExecution`.

CLI syntax

```
aws iot cancel-job-execution \
--job-id <value> \
--thing-name <value> \
[--force | --no-force] \
[--expected-version <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
```

```
"jobId": "string",
"thingName": "string",
"force": boolean,
"expectedVersion": long,
"statusDetails": {
"string": "string"
}
}
```

Pour de plus amples informations, veuillez consulter `cancel-job-execution`.

CreateJob

Crée une tâche. Vous pouvez fournir le document de tâche comme lien vers un fichier dans un compartiment S3 (paramètre `documentSource`) ou dans le corps de la demande (paramètre `document`).

Une tâche peut être rendue continue en définissant le paramètre facultatif `targetSelection` sur `CONTINUOUS` (la valeur par défaut est `SNAPSHOT`). Une tâche continue peut être utilisée pour intégrer ou mettre à niveau des appareils lorsqu'ils sont ajoutés à un groupe, car celui-ci continue de fonctionner et est lancé sur des éléments récemment ajoutés. Cela peut se produire même une fois que les éléments du groupe au moment de la création de la tâche ont terminé la tâche.

Une tâche peut avoir une option TimeoutConfigqui définit la valeur du chronomètre en cours. Le minuteur d'avancement ne peut pas être mis à jour et s'applique à toutes les exécutions de la tâche.

Les validations suivantes sont effectuées sur les arguments de l'API `CreateJob` :

- L'`targets`argument doit être une liste d'objets ou de groupes d'objets valides ARNs. Toutes les choses et tous les groupes d'objets doivent se trouver dans votre Compte AWS.
- L'argument `documentSource` doit être une URL Amazon S3 valide pour un document de tâche. Amazon S3 URLs se présente sous la forme :`https://s3.amazonaws.com/`*`bucketName`*`/`*`objectName`*.
- Le document stocké dans l'URL spécifiée par l'argument `documentSource` doit être un document JSON codé en UTF-8.
- La taille d'un document de tâche est limité à 32 Ko en raison de la limite de la taille d'un message MQTT (128 Ko) et du chiffrement.
- Ils `jobId` doivent être uniques dans votre Compte AWS.

HTTPS request

```
PUT /jobs/jobId

{
"targets": [ "string" ],
"document": "string",
"documentSource": "string",
"description": "string",
"jobTemplateArn": "string",
"presignedUrlConfigData": {
    "roleArn": "string",
    "expiresInSec": "integer"
},
"targetSelection": "CONTINUOUS|SNAPSHOT",
"jobExecutionsRolloutConfig": {
    "exponentialRate": {
       "baseRatePerMinute": integer,
       "incrementFactor": integer,
       "rateIncreaseCriteria": {
          "numberOfNotifiedThings": integer, // Set one or the other
          "numberOfSucceededThings": integer // of these two values.
       },
       "maximumPerMinute": integer
  }
},
"abortConfig": {
   "criteriaList": [
      {
         "action": "string",
         "failureType": "string",
         "minNumberOfExecutedThings": integer,
         "thresholdPercentage": integer
      }
   ]
},
"SchedulingConfig": {
    "startTime": string
    "endTime": string
    "timeZone": string


    "endTimeBehavior": string
```

```
   }
"timeoutConfig": {
  "inProgressTimeoutInMinutes": long
}
}
```

Pour de plus amples informations, veuillez consulter `CreateJob`.

CLI syntax

```
aws iot create-job \
    --job-id <value> \
    --targets <value> \
    [--document-source <value>] \
    [--document <value>] \
    [--description <value>] \
    [--job-template-arn <value>] \
    [--presigned-url-config <value>] \
    [--target-selection <value>] \
    [--job-executions-rollout-config <value>] \
    [--abort-config <value>] \
    [--timeout-config <value>] \
    [--document-parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
    "jobId": "string",
    "targets": [ "string" ],
    "documentSource": "string",
    "document": "string",
    "description": "string",
    "jobTemplateArn": "string",
    "presignedUrlConfig": {
        "roleArn": "string",
        "expiresInSec": long
     },
    "targetSelection": "string",
    "jobExecutionsRolloutConfig": {
          "exponentialRate": {
              "baseRatePerMinute": integer,
```

```
                "incrementFactor": integer,
                "rateIncreaseCriteria": {
                   "numberOfNotifiedThings": integer, // Set one or the other
                   "numberOfSucceededThings": integer // of these two values.
                },
      "maximumPerMinute": integer
      }
    },
    "abortConfig": {
    "criteriaList": [
        {
           "action": "string",
           "failureType": "string",
           "minNumberOfExecutedThings": integer,
           "thresholdPercentage": integer
         }
      ]
    },
    "timeoutConfig": {
          "inProgressTimeoutInMinutes": long
    },
    "documentParameters": {
    "string": "string"
    }
}
```

Pour de plus amples informations, veuillez consulter `create-job`.

DeleteJob

Supprime une tâche et ses exécutions de tâche associées.

Selon le nombre d'exécutions de tâche créées pour la tâche et divers autres facteurs, la suppression d'une tâche peut prendre du temps. Pendant la suppression de la tâche, l'état de celle-ci indique « DELETION_IN_PROGRESS ». Toute tentative de suppression ou d'annulation d'une tâche dont le statut est « DELETION_IN_PROGRESS » entraîne une erreur.

HTTPS request

```
DELETE /jobs/jobId?force=force
```

Pour de plus amples informations, veuillez consulter `DeleteJob`.

CLI syntax

```
aws iot  delete-job \
--job-id <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string",
"force": boolean
}
```

Pour de plus amples informations, veuillez consulter `delete-job`.

DeleteJobExecution

Supprime une exécution de tâche.

HTTPS request

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

Pour de plus amples informations, veuillez consulter `DeleteJobExecution`.

CLI syntax

```
aws iot  delete-job-execution \
--job-id <value> \
--thing-name <value> \
--execution-number <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string",
```

```
"thingName": "string",
"executionNumber": long,
"force": boolean
}
```

Pour de plus amples informations, veuillez consulter delete-job-execution.

DescribeJob

Obtient les détails de l'exécution de tâche.

HTTPS request

```
GET /jobs/jobId
```

Pour de plus amples informations, veuillez consulter DescribeJob.

CLI syntax

```
aws iot describe-job \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string"
}
```

Pour de plus amples informations, veuillez consulter describe-job.

DescribeJobExecution

Obtient les détails d'une exécution de tâche. Le statut de l'exécution de tâche doit être `SUCCEEDED` ou `FAILED`.

HTTPS request

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

Pour de plus amples informations, veuillez consulter DescribeJobExecution.

CLI syntax

```
aws iot  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long
}
```

Pour de plus amples informations, veuillez consulter describe-job-execution.

GetJobDocument

Obtient le document de tâche pour une tâche.

> **Note**
>
> L'espace réservé URLs n'est pas remplacé par Amazon S3 présigné URLs dans le document renvoyé. Les pré-signés ne URLs sont générés que lorsque le service AWS IoT Jobs reçoit une demande via MQTT.

HTTPS request

```
GET /jobs/jobId/job-document
```

Pour de plus amples informations, veuillez consulter GetJobDocument.

CLI syntax

```
aws iot get-job-document \
```

```
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string"
}
```

Pour de plus amples informations, veuillez consulter `get-job-document`.

ListJobExecutionsForJob

Obtient la liste des exécutions de tâche d'une tâche.

HTTPS request

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

Pour de plus amples informations, veuillez consulter `ListJobExecutionsForJob`.

CLI syntax

```
aws iot  list-job-executions-for-job \
--job-id <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Pour de plus amples informations, veuillez consulter list-job-executions-for-job.

ListJobExecutionsForThing

Obtient la liste des exécutions de tâche d'un objet.

HTTPS request

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

Pour de plus amples informations, veuillez consulter ListJobExecutionsForThing.

CLI syntax

```
aws iot list-job-executions-for-thing \
--thing-name <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"thingName": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

Pour de plus amples informations, veuillez consulter list-job-executions-for-thing.

ListJobs

Obtient une liste d'emplois dans votre Compte AWS.

HTTPS request

```
GET /jobs?
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thi
```

Pour de plus amples informations, veuillez consulter `ListJobs`.

CLI syntax

```
aws iot list-jobs \
[--status <value>] \
[--target-selection <value>] \
[--max-results <value>] \
[--next-token <value>] \
[--thing-group-name <value>] \
[--thing-group-id <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"status": "string",
"targetSelection": "string",
"maxResults": "integer",
"nextToken": "string",
"thingGroupName": "string",
"thingGroupId": "string"
}
```

Pour de plus amples informations, veuillez consulter `list-jobs`.

UpdateJob

Met à jour les champs pris en charge de la tâche spécifiée. Les valeurs mises à jour de `timeoutConfig` ne prennent effet que pour les lancements nouvellement en cours. Actuellement, les lancements en cours continuent à être lancés avec la configuration de temporisation précédente.

HTTPS request

```
PATCH /jobs/jobId
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
```

```
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     },
  "maximumPerMinute": number
  },
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

Pour de plus amples informations, veuillez consulter `UpdateJob`.

CLI syntax

```
aws iot  update-job \
--job-id <value> \
[--description <value>] \
[--presigned-url-config <value>] \
[--job-executions-rollout-config <value>] \
[--abort-config <value>] \
[--timeout-config <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
```

```
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     }
  },
  "maximumPerMinute": number
},
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

Pour de plus amples informations, veuillez consulter [update-job](#).

Opérations des API MQTT et HTTPS de l’appareil et types de données

Les commandes suivantes sont disponibles sur les protocoles HTTPS et MQTT. Utilisez ces opérations d’API sur le plan de données pour les appareils exécutant les tâches.

Types de données MQTT et HTTPS pour les appareils de tâches

Les types de données suivants sont utilisés pour communiquer avec le service AWS IoT Jobs via les protocoles MQTT et HTTPS.

JobExecution

L'objet `JobExecution` représente l'exécution d'une tâche sur un appareil. L'exemple suivant montre la syntaxe :

> **Note**
>
> Lorsque vous utilisez les opérations de l'API du plan de données MQTT et HTTP, le type de données `JobExecution` contient un champ `JobDocument`. Vos appareils peuvent utiliser ces informations pour récupérer le document de tâche à partir d'une exécution de tâche.

```
{
    "jobId" : "string",
    "thingName" : "string",
    "jobDocument" : "string",
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
    },
    "queuedAt" : "timestamp",
    "startedAt" : "timestamp",
    "lastUpdatedAt" : "timestamp",
    "versionNumber" : "number",
    "executionNumber": long
}
```

Pour plus d'informations, consultez `JobExecution` ou `job-execution`.

JobExecutionState

`JobExecutionState` contient des informations sur le statut d'exécution d'une tâche. L'exemple suivant montre la syntaxe :

```
{
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
    "statusDetails": {
        "string": "string"
        ...
    }
```

```
    "versionNumber": "number"
}
```

Pour plus d'informations, consultez `JobExecutionState` ou `job-execution-state`.

JobExecutionSummary

Contient un sous-ensemble d'informations sur une exécution de tâche. L'exemple suivant montre la syntaxe :

```
{
    "jobId": "string",
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "versionNumber": "number",
    "executionNumber": long
}
```

Pour plus d'informations, consultez `JobExecutionSummary` ou `job-execution-summary`.

Pour en savoir plus sur les opérations des API MQTT et HTTPS, veuillez consulter les sections suivantes :

- Opérations de l'API MQTT de l'appareil des tâches
- API HTTP de l'appareil de tâches

Opérations de l'API MQTT de l'appareil des tâches

Vous pouvez émettre des commandes sur les appareils de tâches en publiant des messages MQTT dans les rubriques réservées utilisées pour les commandes de tâches.

Le client côté appareil doit être abonné aux sujets des messages de réponse de ces commandes. Si vous utilisez le client de l' AWS IoT appareil, votre appareil s'abonne automatiquement aux sujets de réponse. Cela signifie que l'agent de messages publiera les sujets des messages de réponse à l'intention du client qui a publié le message de commande, que votre client soit abonné ou non aux sujets des messages de réponse. Ces messages de réponse ne passent pas par l'agent de messages et ne peuvent pas être souscrits par d'autres clients ou règles.

Lorsque vous vous abonnez aux rubriques relatives aux tâches et aux événements `jobExecution` de votre solution de surveillance de flotte, activez d'abord les événements relatifs aux tâches

et à l'exécution des tâches pour recevoir tous les événements du côté cloud. Les messages de progression des tâches traités par le biais de l'agent de messages et pouvant être utilisés par les règles AWS IoT sont publiés sous le format Événements Jobs. Étant donné que l'agent de messages publie des messages de réponse, même sans abonnement explicite, votre client doit être configuré pour recevoir et identifier les messages qu'il reçoit. Votre client doit également confirmer que le *thingName* sujet du message entrant s'applique au nom de l'objet du client avant que celui-ci ne donne suite au message.

> **Note**
>
> Les messages envoyés AWS IoT en réponse aux messages de commande de l'API MQTT Jobs sont débités de votre compte, que vous y soyez abonné explicitement ou non.

Ce qui suit montre les opérations de l'API MQTT ainsi que leur syntaxe de demande et de réponse. Toutes les opérations de l'API MQTT sont dotées des paramètres suivants :

clientToken

Une jeton client facultatif utilisé pour établir une corrélation entre les demandes et les réponses. Saisissez une valeur arbitraire ici et elle sera reflétée dans la réponse.

`timestamp`

La durée écoulée, en secondes depuis la date epoch Unix où le message a été envoyé.

GetPendingJobExecutions

Obtient la liste de tous les travaux qui ne sont pas dans un état terminal, pour une chose spécifiée.

Pour appeler cette API, publiez un message sur `$aws/things/`*thingName*`/jobs/get`.

Charge utile de la demande :

```
{ "clientToken": "string" }
```

L'agent de messages publiera `$aws/things/`*thingName*`/jobs/get/accepted` et `$aws/things/`*thingName*`/jobs/get/rejected` même sans abonnement spécifique. Toutefois, pour que votre client reçoive les messages, il doit être à leur écoute. Pour plus d'informations, veuillez consulter la note sur les messages de l'API des tâches.

Charge utile de la réponse :

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ],
"timestamp" : 1489096425069,
"clientToken" : "client-001"
}
```

Où `inProgressJobs` et `queuedJobs` renvoie une liste d'objets JobExecutionSummary dont le statut est `IN_PROGRESS` ou`QUEUED`.

StartNextPendingJobExecution

Obtient et démarre l'exécution de tâche en attente suivante pour un objet (état `IN_PROGRESS` ou `QUEUED`).

- Toutes les exécutions de tâche avec le statut `IN_PROGRESS` sont renvoyées en premier.
- Les exécutions de tâches sont renvoyées dans l'ordre dans lequel ils ont été mis en file d'attente. Lorsqu'un élément est ajouté ou supprimé du groupe cible de votre tâche, confirmez l'ordre de déploiement de toutes les nouvelles exécutions de tâches par rapport aux exécutions de tâches existantes.
- Si la prochaine exécution de tâche en attente est `QUEUED`, son état est modifié en `IN_PROGRESS` et les détails du statut de l'exécution de la tâche sont définis comme indiqué.
- Si la prochaine exécution de tâche en attente est déjà `IN_PROGRESS`, les informations détaillées de son statut ne sont pas modifiées.
- Si aucune exécution de tâche n'est en attente, la réponse n'inclut pas le champ `execution`.
- Le cas échéant, vous pouvez créer un minuteur d'étape en définissant une valeur pour la propriété `stepTimeoutInMinutes`. Si vous ne mettez pas à jour la valeur de cette propriété en exécutant `UpdateJobExecution`, l'exécution de la tâche expire lorsque le minuteur d'étape expire.

Pour appeler cette API, publiez un message sur `$aws/things/`*`thingName`*`/jobs/start-next`.

Charge utile de la demande :

```
{
"statusDetails": {
    "string": "job-execution-state"
```

```
    ...
},
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`statusDetails`

Ensemble de paires nom-valeur décrivant le statut de l'exécution de la tâche. Si aucune valeur n'est spécifiée, les informations `statusDetails` demeurent inchangées.

`stepTimeOutInMinutes`

Spécifie la durée pendant laquelle cet appareil doit terminer l'exécution de la tâche. Si le statut d'exécution de tâche n'est pas défini sur un statut de terminal avant que ce minuteur n'expire, ou avant que le minuteur ne soit réinitialisé (en appelant `UpdateJobExecution`, en définissant le statut sur `IN_PROGRESS` et en spécifiant une nouvelle valeur dans le champ `stepTimeoutInMinutes`), l'état de l'exécution de la tâche est automatiquement défini avec la valeur `TIMED_OUT`. La définition du délai d'expiration n'a aucun effet sur le délai d'exécution de la tâche qui peut avoir été spécifié lorsque la tâche a été créée (`CreateJob` à l'aide du champ `timeoutConfig`).

Les valeurs valides pour ce paramètre varient de 1 à 10 080 (1 minute à 7 jours). Une valeur de -1 est également valide et annulera le chronomètre en cours (créé par une utilisation antérieure de UpdateJobExecutionRequest).

L'agent de messages publiera `$aws/things/`*`thingName`*`/jobs/start-next/accepted` et `$aws/things/`*`thingName`*`/jobs/start-next/rejected` même sans abonnement spécifique. Toutefois, pour que votre client reçoive les messages, il doit être à leur écoute. Pour plus d'informations, veuillez consulter la note sur les messages de l'API des tâches.

Charge utile de la réponse :

```
{
"execution" : JobExecutionData,
"timestamp" : timestamp,
"clientToken" : "string"
}
```

Où `execution` est un objet JobExecution. Par exemple :

```
{
"execution" : {
    "jobId" : "022",
    "thingName" : "MyThing",
    "jobDocument" : "< contents of job document >",
    "status" : "IN_PROGRESS",
    "queuedAt" : 1489096123309,
    "lastUpdatedAt" : 1489096123309,
    "versionNumber" : 1,
    "executionNumber" : 1234567890
},
"clientToken" : "client-1",
"timestamp" : 1489088524284,
}
```

DescribeJobExecution

Permet d'obtenir des informations détaillées sur une exécution de tâche.

Vous pouvez définir le `jobId` sur `$next` pour revenir à la prochaine exécution de tâche en attente (avec le statut `IN_PROGRESS` ou `QUEUED`) pour un objet.

Pour appeler cette API, publiez un message sur `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get`.

Charge utile de la demande :

```
{
"jobId" : "022",
"thingName" : "MyThing",
"executionNumber": long,
"includeJobDocument": boolean,
"clientToken": "string"
}
```

`thingName`

Nom de l'objet associé à l'appareil.

`jobId`

Identifiant unique attribué à cette tâche lors de sa création.

Ou utilisez `$next` pour revenir à la prochaine exécution de tâche en attente (avec le statut `IN_PROGRESS` ou `QUEUED`) pour un objet. Dans ce cas, toutes les exécutions de tâche avec le statut `IN_PROGRESS` sont renvoyées en premier. Les exécutions de tâche sont renvoyées dans l'ordre selon lequel elles ont été créées.

`executionNumber`

(Facultatif) Nombre qui identifie une exécution de tâche sur un appareil. S'il n'est pas indiqué, la dernière exécution de tâche est renvoyée.

`includeJobDocument`

(Facultatif) La réponse contient le document de tâche, sauf si la valeur est `false`. L'argument par défaut est `true`.

L'agent de messages publiera `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/accepted` et `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/get/rejected` même sans abonnement spécifique. Toutefois, pour que votre client reçoive les messages, il doit être à leur écoute. Pour plus d'informations, veuillez consulter la note sur les messages de l'API des tâches.

Charge utile de la réponse :

```
{
"execution" : JobExecutionData,
"timestamp": "timestamp",
"clientToken": "string"
}
```

Où `execution` est un objet JobExecution.

UpdateJobExecution

Met à jour le statut d'une exécution de tâche. Le cas échéant, vous pouvez créer un minuteur d'étape en définissant une valeur pour la propriété `stepTimeoutInMinutes`. Si vous ne mettez pas à jour la valeur de cette propriété en exécutant à nouveau `UpdateJobExecution`, l'exécution de la tâche expire lorsque le minuteur d'étape expire.

Pour appeler cette API, publiez un message sur `$aws/things/`*`thingName`*`/jobs/`*`jobId`*`/update`.

Charge utile de la demande :

```
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"executionNumber": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`status`

Le nouveau statut de l'exécution de la tâche (`IN_PROGRESS`, `FAILED`, `SUCCEEDED` ou`REJECTED`). Il doit être spécifié à chaque mise à jour.

`statusDetails`

Ensemble de paires nom-valeur décrivant le statut de l'exécution de la tâche. Si aucune valeur n'est spécifiée, les informations `statusDetails` demeurent inchangées.

`expectedVersion`

Version actuelle attendue de l'exécution de tâche. Sa version est incrémentée à chaque mise à jour de l'exécution de tâche. Si la version de l'exécution des tâches stockée dans le service AWS IoT Jobs ne correspond pas, la mise à jour est rejetée avec une `VersionMismatch` erreur. Un fichier ErrorResponse contenant les données de statut d'exécution de la tâche en cours est également renvoyé. (Il est donc inutile d'effectuer une demande `DescribeJobExecution` distincte pour obtenir les données du statut d'exécution de tâche.)

`executionNumber`

(Facultatif) Nombre qui identifie une exécution de tâche sur un appareil. S'il n'est pas indiqué, la dernière exécution de tâche est utilisée.

`includeJobExecutionState`

(Facultatif) Quand le champ est inclus et a la valeur `true`, la réponse contient le champ `JobExecutionState`. L'argument par défaut est `false`.

`includeJobDocument`

(Facultatif) Quand le champ est inclus et a la valeur `true`, la réponse contient `JobDocument`. L'argument par défaut est `false`.

`stepTimeoutInMinutes`

Spécifie la durée pendant laquelle cet appareil doit terminer l'exécution de la tâche. Si le statut d'exécution de la tâche n'est pas mis dans un statut terminal avant l'expiration de cette temporisation ou avant la réinitialisation de la temporisation, le statut d'exécution de la tâche est mis à `TIMED_OUT`. La définition ou la réinitialisation du délai d'expiration n'a aucun effet sur le délai d'expiration de l'exécution de la tâche qui peut avoir été spécifié lorsque la tâche a été créée.

L'agent de messages publiera `$aws/things/thingName/jobs/jobId/update/accepted` et `$aws/things/thingName/jobs/jobId/update/rejected` même sans abonnement spécifique. Toutefois, pour que votre client reçoive les messages, il doit être à leur écoute. Pour plus d'informations, veuillez consulter la note sur les messages de l'API des tâches.

Charge utile de la réponse :

```
{
"executionState": JobExecutionState,
"jobDocument": "string",
"timestamp": timestamp,
"clientToken": "string"
}
```

`executionState`

Un objet JobExecutionState.

`jobDocument`

Objet de document de tâche.

> **Note**
>
> Dans les réponses MQTT, le `jobDocument` champ est un objet JSON. Dans les réponses HTTP, il s'agit d'une représentation sous forme de chaîne de l'objet JSON.

`timestamp`

La durée écoulée, en secondes depuis la date epoch Unix où le message a été envoyé.

`clientToken`

Jeton client utilisé pour établir une corrélation entre les demandes et les réponses.

Lorsque vous utilisez le protocole MQTT, vous pouvez également effectuer les mises à jour suivantes :

JobExecutionsChanged

Envoyé chaque fois qu'une exécution de tâche est ajoutée à la liste des exécutions de tâche en attente pour un objet, ou en est supprimée.

Utilisez la rubrique :

`$aws/things/thingName/jobs/notify`

Charge utile du message :

```
{
"jobs" : {
    "JobExecutionState": [ JobExecutionSummary ... ]
         },
    "timestamp": timestamp
}
```

NextJobExecutionChanged

Envoyé chaque fois qu'il y a une modification apportée à l'exécution de tâche définie comme exécution suivante dans la liste des exécutions de tâche en attente pour un objet, comme défini pour DescribeJobExecution avec `jobId$next`. Le message n'est pas envoyé en cas de modification des détails de l'exécution de tâche suivante, mais uniquement lorsque la prochaine tâche qui sera renvoyée par `DescribeJobExecution` avec le `jobId $next` a changé. Considérons les exécutions de tâche J1 et J2 avec le statut `QUEUED`. J1 est l'exécution suivante sur la liste des exécutions de tâche en attente. Si le statut de J2 devient `IN_PROGRESS` tandis que l'état de J1 reste inchangé, cette notification est envoyée et contient les détails de J2.

Utilisez la rubrique :

`$aws/things/thingName/jobs/notify-next`

Charge utile du message :

```
{
"execution" : JobExecution,
"timestamp": timestamp,
}
```

API HTTP de l'appareil de tâches

Les appareils peuvent communiquer avec AWS IoT Jobs à l'aide de la signature HTTP version 4 sur le port 443. Il s'agit de la méthode utilisée par la CLI AWS SDKs et. Pour plus d'informations sur ces outils, voir Référence des AWS CLI commandes : iot-jobs-data ou AWS SDKs et Outils.

Les commandes suivantes sont disponibles pour les appareils exécutant les tâches. Pour plus d'informations sur l'utilisation des opérations d'API avec le protocole MQTT, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.

GetPendingJobExecutions

Obtient la liste de tous les travaux qui ne sont pas dans un état terminal, pour une chose spécifiée.

HTTPS request

```
GET /things/thingName/jobs
```

Réponse :

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ]
}
```

Pour de plus amples informations, veuillez consulter `GetPendingJobExecutions`.

CLI syntax

```
aws iot-jobs-data get-pending-job-executions \
--thing-name <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"thingName": "string"
}
```

Pour de plus amples informations, veuillez consulter `get-pending-job-executions`.

StartNextPendingJobExecution

Obtient et démarre l'exécution de tâche en attente suivante pour un objet (avec un statut `IN_PROGRESS` ou `QUEUED`).

- Toutes les exécutions de tâche avec le statut `IN_PROGRESS` sont renvoyées en premier.
- Les exécutions de tâche sont renvoyées dans l'ordre selon lequel elles ont été créées.
- Si la prochaine exécution de tâche en attente est `QUEUED`, son statut est modifié en `IN_PROGRESS` et les détails du statut de l'exécution de la tâche sont définis comme indiqué.
- Si la prochaine exécution de tâche en attente est déjà `IN_PROGRESS`, les informations détaillées de son statut ne sont pas modifiées.
- Si aucune exécution de tâche n'est en attente, la réponse n'inclut pas le champ `execution`.
- Le cas échéant, vous pouvez créer un minuteur d'étape en définissant une valeur pour la propriété `stepTimeoutInMinutes`. Si vous ne mettez pas à jour la valeur de cette propriété en exécutant `UpdateJobExecution`, l'exécution de la tâche expire lorsque le minuteur d'étape expire.

HTTPS request

L'exemple suivant montre la syntaxe de demande :

```
PUT /things/thingName/jobs/$next
{
"statusDetails": {
    "string": "string"
    ...
},
"stepTimeoutInMinutes": long
}
```

Pour de plus amples informations, veuillez consulter `StartNextPendingJobExecution`.

CLI syntax

Résumé :

```
aws iot-jobs-data  start-next-pending-job-execution \
--thing-name <value> \
{--step-timeout-in-minutes <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"thingName": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": long
}
```

Pour de plus amples informations, veuillez consulter `start-next-pending-job-execution`.

DescribeJobExecution

Permet d'obtenir des informations détaillées sur une exécution de tâche.

Vous pouvez définir le `jobId` sur `$next` pour revenir à la prochaine exécution de tâche en attente pour un objet. Le statut de l'exécution de tâche doit être `QUEUED` ou `IN_PROGRESS`.

HTTPS request

Requête :

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

Réponse :

```
{
"execution" : JobExecution,
```

```
}
```

Pour de plus amples informations, veuillez consulter DescribeJobExecution.

CLI syntax

Résumé :

```
aws iot-jobs-data  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string",
"thingName": "string",
"includeJobDocument": boolean,
"executionNumber": long
}
```

Pour de plus amples informations, veuillez consulter describe-job-execution.

UpdateJobExecution

Met à jour le statut d'une exécution de tâche. Le cas échéant, vous pouvez créer un minuteur d'étape en définissant une valeur pour la propriété `stepTimeoutInMinutes`. Si vous ne mettez pas à jour la valeur de cette propriété en exécutant à nouveau `UpdateJobExecution`, l'exécution de la tâche expire lorsque le minuteur d'étape expire.

HTTPS request

Requête :

```
POST /things/thingName/jobs/jobId
{
"status": "job-execution-state",
"statusDetails": {
```

```
    "string": "string"
    ...
},
"expectedVersion": "number",
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"executionNumber": long
}
```

Pour de plus amples informations, veuillez consulter UpdateJobExecution.

CLI syntax

Résumé :

```
aws iot-jobs-data  update-job-execution \
--job-id <value> \
--thing-name <value> \
--status <value> \
[--status-details <value>] \
[--expected-version <value>] \
[--include-job-execution-state | --no-include-job-execution-state] \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--step-timeout-in-minutes <value>] \
[--generate-cli-skeleton]
```

Format `cli-input-json` :

```
{
"jobId": "string",
"thingName": "string",
"status": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": number,
"expectedVersion": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"executionNumber": long
```

```
}
```

Pour de plus amples informations, veuillez consulter `update-job-execution`.

Sécuriser les utilisateurs et les appareils avec AWS IoT Jobs

Pour autoriser les utilisateurs à utiliser AWS IoT des tâches avec leurs appareils, vous devez leur accorder des autorisations en utilisant des politiques IAM. Les appareils doivent ensuite être autorisés en utilisant des AWS IoT Core politiques pour se connecter en toute sécurité AWS IoT, recevoir des exécutions de tâches et mettre à jour l'état d'exécution.

Type de politique requis pour AWS IoT Jobs

Le tableau suivant présente les différents types de stratégies que vous devez utiliser pour l'autorisation. Pour de plus amples informations sur les politiques requises à utiliser, veuillez consulter Autorisation.

Type politique requise.

Cas d'utilisation	Protocol	Authentification	plane/data Plan de contrôle	Type d'identité	Type politique requise.
Autoriser un administrateur, un opérateur ou un service cloud à travailler en toute sécurité avec les tâches	HTTPS	AWS Authentification par signature version 4 (port 443)	Le plan de contrôle et le plan de données.	Amazon Cognito Identity, IAM ou utilisateur fédéré	Politique IAM
Autorisez votre appareil IoT à fonctionn	HTTPS/MQTT	Authentification mutuelle TCP ou TLS (port 8883 ou 443)	Plan de données	Certificats X.509	AWS IoT Core politique

Cas d'utilisation	Protocol	Authentification	plane/data Plan de contrôle	Type d'identité	Type politique requise.
er en toute sécurité avec les tâches					

Pour autoriser les opérations AWS IoT Jobs qui peuvent être effectuées à la fois sur le plan de contrôle et sur le plan de données, vous devez utiliser les politiques IAM. Les identités doivent avoir été authentifiées auprès de AWS IoT pour effectuer ces opérations, qui doivent être Identités Amazon Cognito ouUtilisateurs, groupes et rôles IAM. Pour de plus amples informations sur l'authentification, veuillez consulter Authentification.

Les appareils doivent désormais être autorisés sur le plan de données en utilisant des AWS IoT Core politiques pour se connecter en toute sécurité à la passerelle des appareils. La passerelle permet aux appareils de communiquer en toute sécurité avec les tâches AWS IoT, de recevoir des exécutions de tâches et de mettre à jour l'état d'exécution des tâches. La communication entre les appareils est sécurisée à l'aide de protocoles sécurisés de communication MQTT ou HTTPS. Ces protocoles utilisent Certificats client X.509 ceux fournis par AWS IoT pour authentifier les connexions des appareils.

Voici comment vous autorisez vos utilisateurs, vos services cloud et vos appareils à utiliser AWS IoT Jobs. Pour plus d'informations sur les opérations de plan de contrôle et le plan de données, veuillez consulter le AWS IoT opérations de l'API jobs.

Rubriques

- Autoriser les utilisateurs et les services cloud à utiliser les tâches AWS IoT
- Autoriser vos appareils à utiliser AWS IoT Jobs en toute sécurité sur le plan de données

Autoriser les utilisateurs et les services cloud à utiliser les tâches AWS IoT

Pour autoriser vos utilisateurs et vos services cloud, vous devez utiliser des politiques IAM à la fois sur le plan de contrôle et sur le plan de données. Les politiques doivent être utilisées avec le protocole HTTPS et doivent utiliser l'authentification Signature Version 4 (port 443) AWS pour authentifier les utilisateurs.

> **Note**
>
> AWS IoT Core les politiques ne doivent pas être utilisées sur le plan de contrôle. Seules les politiques IAM sont utilisées pour autoriser les utilisateurs ou les services cloud. Pour de plus amples informations sur les types de politiques requises à utiliser, veuillez consulter Type de politique requis pour AWS IoT Jobs.

Les politiques IAM sont des documents JSON qui contiennent des déclarations de politique. Les déclarations de politique utilisent les éléments Effet, Action et Ressource pour préciser les ressources, les actions autorisées ou rejetées et les conditions dans lesquelles les actions sont autorisées ou rejetées. Pour de plus amples informations, veuillez consulter Références des éléments de stratégie JSON IAM dans le Guide de l'utilisateur IAM.

> **Warning**
>
> Nous vous recommandons de ne pas utiliser d'autorisations génériques, comme `"Action": ["iot:*"]` dans vos politiques ou AWS IoT Core politiques IAM. L'utilisation d'autorisations génériques n'est pas une bonne pratique de sécurité recommandée. Pour plus d'informations, consultez AWS IoT la politique trop permissive.

Politiques IAM sur le plan de contrôle

Sur le plan de contrôle, les politiques IAM utilisent le préfixe `iot:` associé à l'action pour autoriser le fonctionnement de l'API de tâches correspondante. Par exemple, l'action de politique `iot:CreateJob` accorde à l'utilisateur l'autorisation d'utiliser l'API `CreateJob`.

Actions de politique

Le tableau suivant contient les actions de politique IAM et les autorisations d'utilisation des actions d'API. Pour plus d'informations sur les types de ressources, consultez la section Types de ressources définis par AWS IoT. Pour plus d'informations sur AWS IoT les actions, consultez la section Actions définies par AWS IoT.

Actions de politique IAM sur le plan de contrôle

Actions de politique	Opération API	Types de ressources	Description
`iot:AssociateTargetsWithJob`	AssociateTargetsWithJob	• tâche • thing • thinggroup	Représente l'autorisation d'associer un groupe à une tâche continue. L'autorisation `iot:AssociateTargetsWithJob` est vérifiée chaque fois qu'une demande est faite d'associer les cibles.
`iot:CancelJob`	CancelJob	tâche	Représente l'autorisation d'annuler une tâche. L'autorisation `iot:CancelJob` est vérifiée chaque fois qu'une demande est faite d'annuler une tâche.
`iot:CancelJobExecution`	CancelJobExecution	• tâche • thing	Représente l'autorisation d'annuler une exécution de tâche. L'autorisation `iot:CancelJobExecution` est vérifiée chaque fois qu'une demande est faite d'annuler une exécution de tâche.
`iot:CreateJob`	CreateJob	• tâche • thing • thinggroup • modèle de tâche • package	Représente l'autorisation de créer une tâche. L'autorisation `iot: CreateJob` est vérifiée chaque fois qu'une demande est faite de créer une tâche.
`iot:CreateJobTemplate`	CreateJobTemplate	• tâche • modèle de tâche	Représente l'autorisation de créer un modèle de tâche. L'autorisation `iot:CreateJobTemplate` est vérifiée chaque fois qu'une demande est faite de créer un modèle de tâche.

Actions de politique	Opération API	Types de ressources	Description
		• package	
iot:DeleteJob	DeleteJob	tâche	Représente l'autorisation de supprimer une tâche. L'autorisation iot: DeleteJob est vérifiée chaque fois qu'une demande est faite de supprimer une tâche.
iot:DeleteJobTemplate	DeleteJobTemplate	modèle de tâche	Représente l'autorisation de supprimer un modèle de tâche. L'autorisation iot: CreateJobTemplate est vérifiée chaque fois qu'une demande est faite de supprimer un modèle de tâche.
iot:DeleteJobExecution	DeleteJobTemplate	• tâche • thing	Représente l'autorisation de supprimer une exécution de tâche. L'autorisation iot: DeleteJobExecution est vérifiée chaque fois qu'une demande est faite de supprimer une exécution de tâche.
iot:DescribeJob	DescribeJob	tâche	Représente l'autorisation de décrire une tâche. L'autorisation iot: DescribeJob est vérifiée chaque fois qu'une demande est faite de décrire une tâche.
iot:DescribeJobExecution	DescribeJobExecution	• tâche • thing	Représente l'autorisation de décrire une exécution de tâche. L'autorisation iot: DescribeJobExecution est vérifiée chaque fois qu'une demande est faite de décrire une exécution de tâche.

Actions de politique	Opération API	Types de ressources	Description
`iot:DescribeJobTemplate`	DescribeJobTemplate	modèle de tâche	Représente l'autorisation de décrire un modèle de tâche. L'autorisation `iot:DescribeJobTemplate` est vérifiée chaque fois qu'une demande est faite de décrire un modèle de tâche.
`iot:DescribeManagedJobTemplate`	DescribeManagedJobTemplate	modèle de tâche	Représente l'autorisation de décrire un modèle de tâche gérée. L'autorisation `iot: DescribeManagedJobTemplate` est vérifiée chaque fois qu'une demande est faite de décrire un modèle de tâche gérée.
`iot:GetJobDocument`	GetJobDocument	tâche	Représente l'autorisation d'obtenir le document de tâche correspondant à une tâche. L'autorisation `iot:GetJobDocument` est vérifiée chaque fois qu'une demande est faite d'obtenir un document de tâche.
`iot:ListJobExecutionsForJob`	ListJobExecutionsForJob	tâche	Représente l'autorisation de répertorier les exécutions de tâche d'une tâche. L'autorisation `iot:ListJobExecutionsForJob` est vérifiée chaque fois qu'une demande est faite de répertorier les exécutions d'une tâche.

Actions de politique	Opération API	Types de ressources	Description
iot:ListJobExecutionsForThing	ListJobExecutionsForThing	thing	Représente l'autorisation de répertorier les exécutions de tâche d'une tâche. L'autorisation iot:ListJobExecutionsForThing est vérifiée chaque fois qu'une demande est faite de répertorier les exécutions d'un objet.
iot:ListJobs	ListJobs	none	Représente l'autorisation de répertorier les tâches. L'autorisation iot:ListJobs est vérifiée chaque fois qu'une demande est faite de répertorier les tâches.
iot:ListJobTemplates	ListJobTemplates	none	Représente l'autorisation de répertorier les modèles de tâche. L'autorisation iot:ListJobTemplates est vérifiée chaque fois qu'une demande est faite répertorier les modèles de tâche.
iot:ListManagedJobTemplates	ListManagedJobTemplates	none	Représente l'autorisation de répertorier les modèles de tâches gérées. L'autorisation iot:ListManagedJobTemplates est vérifiée chaque fois qu'une demande est faite répertorier les modèles de tâche gérée.
iot:UpdateJob	UpdateJob	tâche	Représente l'autorisation de mettre à jour une tâche. L'autorisation iot:UpdateJob est vérifiée chaque fois qu'une demande est faite de mettre une tâche.

Actions de politique	Opération API	Types de ressources	Description
`iot:TagResource`	TagResource	• tâche • modèle de tâche • thing	Octroie l'autorisation de baliser une ressource spécifique.
`iot:UntagResource`	UntagResource	• tâche • modèle de tâche • thing	Octroie l'autorisation d'annuler le balisage d'une ressource spécifique.

Exemple de politique IAM de base

L'exemple suivant illustre une politique IAM qui autorise l'utilisateur à effectuer les actions suivantes pour votre objet et groupe d'objets IoT.

Dans l'exemple, remplacez :

- *region*avec votre Région AWS, par exemple`us-east-1`.
- *account-id*avec votre Compte AWS numéro, par exemple`57EXAMPLE833`.
- *thing-group-name*avec le nom de votre groupe d'objets IoT pour lequel vous ciblez des emplois, tel que`FirmwareUpdateGroup`.
- *thing-name*avec le nom de l'objet IoT pour lequel vous ciblez des emplois, par exemple`MyIoTThing`.

JSON

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

```
    "Statement": [
        {
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob",
                ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:thinggroup/thing-group-
name"
        },
        {
            "Action": [
                "iot:DescribeJob",
                "iot:CancelJob",
                "iot:DeleteJob",
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:job/*"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iot:region:account-id:thing/thing-name"
                "arn:aws:iot:region:account-id:job/*"
            ]
        }
    ]
}
```

Exemple de politique IAM pour l'autorisation basée sur IP

Vous pouvez empêcher les principaux d'effectuer des appels d'API vers le point de terminaison de votre plan de contrôle à partir d'adresses IP spécifiques. Pour spécifier les adresses IP qui peuvent être autorisées, dans l'élément Condition de votre politique IAM, utilisez la clé de condition globale `aws:SourceIp`.

L'utilisation de cette clé de condition peut également empêcher d'autres Service AWS personnes d'effectuer ces appels d'API en votre nom, par exemple AWS CloudFormation. Pour autoriser l'accès à ces services, utilisez la clé de condition `aws:ViaAWSService`globale associée à la SourceIp clé aws :. Cela garantit que la restriction d'accès à l'adresse IP source s'applique uniquement aux requêtes faites directement par un principal. Pour plus d'informations, voir AWS: Refuse l'accès à AWS en fonction de l'adresse IP source.

L'exemple suivant montre comment autoriser uniquement une adresse IP spécifique qui peut effectuer des appels d'API vers le point de terminaison du plan de contrôle. La clé `aws:ViaAWSService` est définie sur `true`, ce qui permet aux autres services d'effectuer des appels d'API en votre nom.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
             ],
            "Resource": ["*"],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "123.45.167.89"
                }
            },
            "Bool": {"aws:ViaAWSService": "true"}
        }
    ],
}
```

Politiques IAM sur le plan de données

Les politiques IAM sur le plan de données utilisent le préfixe `iotjobsdata:` pour autoriser les opérations d'API de tâches que les utilisateurs peuvent effectuer. Sur le plan de données, vous

pouvez autoriser un utilisateur à utiliser l'API `DescribeJobExecution` en utilisant l'action de politique `iotjobsdata:DescribeJobExecution`.

> **Warning**
>
> L'utilisation de politiques IAM sur le plan de données n'est pas recommandée lorsque vous ciblez des tâches AWS IoT pour vos appareils. Nous vous recommandons d'utiliser des politiques IAM sur le plan de contrôle pour que les utilisateurs puissent créer et gérer des tâches. Sur le plan de données, pour autoriser les appareils à récupérer les exécutions de tâches et à mettre à jour le statut d'exécution, utilisez AWS IoT Core politiques pour le protocole HTTPS.

Exemple de politique IAM de base

Les opérations d'API qui doivent être autorisées sont généralement effectuées en tapant des commandes CLI. L'exemple suivant un exemple d'un utilisateur exécutant une opération `DescribeJobExecution`.

Dans l'exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`us-east-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple`57EXAMPLE833`.
- *`thing-name`*avec le nom de l'objet IoT pour lequel vous ciblez des emplois, par exemple`myRegisteredThing`.
- *`job-id`* est l'identifiant unique de la tâche ciblée à l'aide de l'API.

```
aws iot-jobs-data describe-job-execution \
    --endpoint-url "https://account-id.jobs.iot.region.amazonaws.com" \
    --job-id jobID --thing-name thing-name
```

Un exemple de politique IAM autorisant cette action est présenté ci-dessous :

JSON

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

```
    "Statement":
    {
        "Action": ["iotjobsdata:DescribeJobExecution"],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:region:account-id:thing/thing-name",
    }
   }
```

Exemples de politiques IAM pour l'autorisation basée sur IP

Vous pouvez empêcher les principaux d'effectuer des appels d'API vers le point de terminaison de votre plan de données à partir d'adresses IP spécifiques. Pour spécifier les adresses IP qui peuvent être autorisées, dans l'élément Condition de votre politique IAM, utilisez la clé de condition globale`aws:SourceIp`.

L'utilisation de cette clé de condition peut également empêcher d'autres Service AWS personnes d'effectuer ces appels d'API en votre nom, par exemple AWS CloudFormation. Pour autoriser l'accès à ces services, utilisez la clé de condition globale `aws:ViaAWSService` avec la clé de condition `aws:SourceIp`. Cela garantit que la restriction d'accès à l'adresse IP ne s'applique qu'aux demandes directement effectuées par le principal. Pour plus d'informations, voir AWS: Refuse l'accès à AWS en fonction de l'adresse IP source.

L'exemple suivant montre comment autoriser uniquement une adresse IP spécifique qui peut effectuer des appels d'API vers le point de terminaison du plan de données.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
            "Effect": "Allow",
            "Action": ["iotjobsdata:*"],
            "Resource": ["*"],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "123.45.167.89"
                }
            },
            "Bool": {"aws:ViaAWSService": "false"}
```

```
        }
    ],
}
```

L'exemple suivant montre comment empêcher des adresses IP ou des plages d'adresses spécifiques d'effectuer des appels d'API vers le point de terminaison du plan de données.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
            "Effect": "Deny",
            "Action": ["iotjobsdata:*"],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "123.45.167.89",
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            },
            "Resource": ["*"],
        }
    ],
}
```

Exemple de politique IAM pour le plan de contrôle et le plan de données

Si vous effectuez une opération d'API à la fois sur le plan de contrôle et sur le plan de données, votre action de politique du plan de contrôle doit utiliser le préfixe `iot:`, et votre action de stratégie de plan de données doit utiliser le préfixe `iotjobsdata:`.

Par exemple, l'API `DescribeJobExecution` peut être utilisée à la fois dans le plan de contrôle et dans le plan de données. Sur le plan de contrôle, l'DescribeJobExecutionAPI est utilisée pour décrire l'exécution d'une tâche. Sur le plan de données, l' DescribeJobExecutionAPI est utilisée pour obtenir les détails de l'exécution d'une tâche.

La politique IAM suivante autorise un utilisateur à utiliser l'API `DescribeJobExecution` à la fois sur le plan de contrôle et sur le plan de données.

Dans l'exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`us-east-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple`57EXAMPLE833`.
- *`thing-name`*avec le nom de l'objet IoT pour lequel vous ciblez des emplois, par exemple`MyIoTThing`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["iotjobsdata:DescribeJobExecution"],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:thing/thing-name"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iot:region:account-id:thing/thing-name"
                "arn:aws:iot:region:account-id:job/*"
            ]
        }
    ]
}
```

Autoriser le balisage des ressources IoT

Pour mieux contrôler les tâches et les modèles de tâches que vous pouvez créer, modifier ou utiliser, vous pouvez associer des balises aux tâches ou aux modèles de tâches. Les balises vous aident

également à identifier les propriétaires et à attribuer et répartir les coûts en les plaçant dans des groupes de facturation et en y attachant des balises.

Lorsqu'un utilisateur souhaite baliser ses tâches ou les modèles de tâches qu'il a créés à l'aide du AWS Management Console ou du AWS CLI, votre politique IAM doit autoriser l'utilisateur à les baliser. Pour accorder des autorisations, votre politique IAM doit utiliser l'action `iot:TagResource`.

> **Note**
>
> Si votre politique IAM n'inclut pas l'action `iot:TagResource`, toute `CreateJob` ou `CreateJobTemplate`contenant une balise renverra une erreur `AccessDeniedException`.

Lorsque vous souhaitez étiqueter vos tâches ou les modèles de tâches que vous avez créés à l'aide du AWS Management Console ou du AWS CLI, votre politique IAM doit autoriser ces tâches. Pour accorder des autorisations, votre politique IAM doit utiliser l'action `iot:TagResource`.

Pour obtenir des informations générales sur le balisage des ressources, veuillez consulter Marquer vos ressources AWS IoT.

Exemple de politique IAM

Reportez-vous aux exemples de politique IAM suivants octroyant des autorisations de balisage :

Exemple 1

Utilisateur qui exécute la commande suivante pour créer une tâche et l'associer à un environnement spécifique.

Dans cet exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`us-east-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple`57EXAMPLE833`.
- *`thing-name`*avec le nom de l'objet IoT pour lequel vous ciblez des emplois, par exemple`MyIoTThing`.

```
aws iot create-job
    --job-id test_job
    --targets "arn:aws:iot:region:account-id:thing/thingOne"
    --document-source "https://s3.amazonaws.com/amzn-s3-demo-bucket/job-document.json"
```

```
    --description "test job description"
    --tags Key=environment,Value=beta
```

Pour cet exemple, vous devez utiliser la politique IAM suivante :

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": [ "iot:CreateJob", "iot:CreateJobTemplate",
 "iot:TagResource" ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:iot:aws-region:account-id:job/*",
            "arn:aws:iot:aws-region:account-id:jobtemplate/*"
        ]
    }
}
```

Autoriser vos appareils à utiliser AWS IoT Jobs en toute sécurité sur le plan de données

Pour autoriser vos appareils à interagir en toute sécurité avec AWS IoT Jobs sur le plan de données, vous devez utiliser AWS IoT Core des politiques. AWS IoT Core les politiques pour les tâches sont des documents JSON contenant des déclarations de politique. Ces politiques utilisent également des éléments Effet, Action et Ressource, et suivent une convention similaire à celle des politiques IAM. Pour de plus amples informations sur les éléments, veuillez consulter Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Les politiques peuvent être utilisées avec les protocoles MQTT et HTTPS et doivent utiliser l'authentification mutuelle TCP ou TLS pour authentifier les appareils. L'exemple suivant montre comment utiliser ces politiques dans les différents protocoles de communication.

> **Warning**
>
> Nous vous recommandons de ne pas utiliser d'autorisations génériques, comme `"Action": ["iot:*"]` dans vos politiques ou AWS IoT Core politiques IAM. L'utilisation d'autorisations

génériques n'est pas une bonne pratique de sécurité recommandée. Pour plus d'informations, consultez AWS IoT la politique trop permissive.

AWS IoT Core politiques pour le protocole MQTT

AWS IoT Core les politiques du protocole MQTT vous accordent l'autorisation d'utiliser les actions de l'API MQTT du dispositif de tâches. Les opérations de l'API MQTT sont utilisées pour travailler avec des rubriques MQTT réservées aux commandes de tâches. Pour plus d'informations sur ces opérations d'API, veuillez consulter Opérations de l'API MQTT de l'appareil des tâches.

Les politiques MQTT utilisent des actions politiques telles que `iot:Connect`, `iot:Publish`, `iot:Subscribe` et `iot:Receieve` pour travailler avec les rubriques des tâches. Ces politiques vous permettent de vous connecter à l'agent de messages, de vous abonner aux sujets MQTT des jobs et d'envoyer et de recevoir des messages MQTT entre vos appareils et le cloud. Pour plus d'informations sur ces actions, consultez AWS IoT Core actions politiques.

Pour plus d'informations sur les rubriques relatives aux AWS IoT offres d'emploi, consultezRubriques de tâche.

Exemple de politique MQTT de base

L'exemple suivant montre comment vous pouvez utiliser `iot:Publish` et `iot:Subscribe` pour publier et vous abonner à des tâches et à des exécutions de tâches.

Dans l'exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`us-east-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple`57EXAMPLE833`.
- *`thing-name`*avec le nom de l'objet IoT pour lequel vous ciblez des emplois, par exemple`MyIoTThing`.

JSON

```
{
    "Statement": [
        {
           "Effect": "Allow",
```

```
            "Action": [
                "iot:Publish",
                "iot:Subscribe"
             ],
             "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/events/job/*",
                "arn:aws:iot:region:account-id:topic/$aws/events/jobExecution/*",
                "arn:aws:iot:region:account-id:topic/$aws/things/thing-name/jobs/
*"
             ]
         }
    ],
    "Version": "2012-10-17"
}
```

AWS IoT Core politiques pour le protocole HTTPS

AWS IoT Core les politiques du plan de données peuvent également utiliser le protocole HTTPS avec le mécanisme d'authentification TLS pour autoriser vos appareils. Sur le plan de données, les politiques utilisent le préfixe `iotjobsdata:` pour autoriser les opérations d'API de tâches que vos appareils peuvent effectuer. Par exemple, l'action de politique `iotjobsdata:DescribeJobExecution` accorde à l'utilisateur l'autorisation d'utiliser l'API `DescribeJobExecution`.

> **Note**
>
> Les actions de politique du plan de données doivent utiliser le préfixe `iotjobsdata:`. Sur le plan de contrôle, les actions doivent utiliser le préfixe `iot:`. Pour un exemple de politique IAM lorsque des actions de politique du plan de contrôle et du plan de données sont utilisées à la fois, veuillez consulter Exemple de politique IAM pour le plan de contrôle et le plan de données.

Actions de politique

Le tableau suivant présente une liste des actions AWS IoT Core politiques et des autorisations permettant d'autoriser les appareils à utiliser les actions de l'API. Pour obtenir la liste des opérations d'API que vous pouvez effectuer dans le plan de données, veuillez consulter API HTTP de l'appareil de tâches.

> **Note**
>
> Ces actions de stratégie d'exécution de tâche s'appliquent uniquement pour le point de terminaison HTTP TLS. Si vous utilisez le point de terminaison MQTT, vous devez utiliser les actions de politique MQTT précédemment.

AWS IoT Core actions politiques sur le plan de données

Actions de politique	Opération API	Types de ressources	Description
`iotjobsdata:DescribeJobExecution`	DescribeJobExecution	• tâche • thing	Représente l'autorisation de récupérer une exécution de tâch. L'autorisation `iotjobsdata:DescribeJobExecution` est vérifiée chaque fois qu'une demande est faite de récupérer une exécution de tâche.
`iotjobsdata:GetPendingJobExecutions`	GetPendingJobExecutions	thing	Représente l'autorisation de récupérer la liste des tâches qui ne sont pas à un statut terminal pour un objet. L'autorisation `iotjobsdata:GetPendingJobExecutions` est vérifiée chaque fois qu'une demande est faite de récupérer la liste.
`iotjobsdata:StartNextPendingJobExecution`	StartNextPendingJobExecution	thing	Représente l'autorisation d'obtenir et de démarrer l'exécution de tâche en attente suivante pour un objet. C'est-à-dire de mettre à jour une exécution de tâche en la faisant passer du statut `QUEUED` au statut `IN_PROGRESS` .) L'autorisation `iotjobsdata:StartNextPendingJobExecution` est

Actions de politique	Opération API	Types de ressources	Description
			vérifiée chaque fois qu'une demande est faite de démarrer l'exécution de tâche en attente suivante.
`iotjobsdata:UpdateJobExecution`	UpdateJobExecution	thing	Représente l'autorisation de mettre à jour une exécution de tâche. L'autorisation `iotjobsdata:UpdateJobExecution` est vérifiée chaque fois qu'une demande est faite de mettre à jour l'état d'une exécution de tâche.

Exemple de politique de base

Vous trouverez ci-dessous un exemple de AWS IoT Core politique qui autorise l'exécution des actions sur les opérations de l'API du plan de données pour n'importe quelle ressource. Vous pouvez étendre votre politique à une ressource spécifique, telle qu'un objet IoT. Dans votre exemple, remplacez :

- *region*avec votre Région AWS tel que`us-east-1`.
- *account-id*avec votre Compte AWS numéro, par exemple`57EXAMPLE833`.
- *thing-name*avec le nom de l'IoT, tel que`MyIoTthing`.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
```

```
                "iotjobsdata:DescribeJobExecution",
                "iotjobsdata:UpdateJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:thing/thing-name"
        }
    ]
}
```

Par exemple, vous devez utiliser ces politiques lorsque vos appareils IoT utilisent une politique AWS IoT Core pour accéder à l'une de ces opérations d'API, comme dans l'exemple d'API `DescribeJobExecution` suivant :

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument&namespaceId=namespaceId
 HTTP/1.1
```

AWS IoT Limites d'emplois

AWS IoT Jobs est soumis à des quotas de service, ou limites, qui correspondent au nombre maximum de ressources de service ou d'opérations pour vous Compte AWS.

Rubriques

- Limites relatives à l'exécution des tâches
- Limites de tâches actives et simultanées

Limites relatives à l'exécution des tâches

Cette section fournit des informations sur les limites d'exécution des tâches pour AWS IoT Device Management.

> **Note**
>
> Ces limites ne font pas partie des quotas de service que vous trouverez dans la documentation relative aux Quotas de AWS IoT Device Management service.

Pour obtenir des informations sur le nombre d'exécutions de tâches en attente, vous pouvez soit utiliser l'`GetPendingJobExecutions`API, soit vous abonner aux rubriques MQTT réservées aux AWS IoT tâches et à la réceptionType de notification de tâche..

Le nombre d'exécutions de tâches en attente dans votre compte peut varier selon que vous avez activé la configuration de planification et que vous utilisez une fenêtre de maintenance récurrente.

Nombre maximum d'exécutions de tâches en attente

Nom de l'API/ de la notification	Description	Sans planification de configuration	Avec configuration de planification
`ListNotification`	A `ListNotification` est publié chaque fois qu'une ancienne exécution de tâche passe au statut de terminal, ou lorsqu'une nouvelle exécution de tâche est mise en file d'attente ou passe à un statut non terminal. Il peut afficher jusqu'à 15 exécutions de tâches en attente qui sont `QUEUED` soit`IN_PROGRESS` .	10	15 (Jusqu'à 5 exécutions de tâches n'apparaissent que `ListNotification` pendant une fenêtre de maintenance).
`GetPendingJobExecutions`	Lorsque vous appelez l'`GetPendingJobExecutions` API, elle renvoie une liste des exécutions de tâches qui n'ont pas encore commencé et qui peuvent être démarrées après l'appel d'API. L'API peut renvoyer jusqu'à 10 exécutions de tâches en attente. • Sur les 10 exécutions de tâches en attente, les exécutions `IN_PROGRESS` seront filtrées du résultat. • Sur les 10 exécutions de tâches en attente, si leurs tâches sont en `SCHEDULED` état, elles seront filtrées des résultats.	10	15

Limites de tâches actives et simultanées

Cette section vous aidera à en savoir plus sur les tâches actives et simultanées ainsi que sur les limites qui s'y appliquent.

Tâches actives et limite de tâches actives

Lorsque vous créez une tâche à l'aide de la AWS IoT console ou de l'`CreateJob`API, le statut de la tâche devient`IN_PROGRESS`. Toutes les tâches en cours sont des tâches actives et sont prises en compte dans le calcul de la limite des tâches actives. Cela inclut les tâches qui déploient de nouvelles exécutions de tâches ou qui attendent que les appareils terminent leur exécution. Cette limite s'applique à la fois aux tâches continues et aux tâches instantanées.

Tâches simultanées et limite de simultanéité des tâches

Les tâches en cours qui déploient de nouvelles exécutions de tâches ou annulent des exécutions de tâches créées précédemment sont des tâches simultanées et sont prises en compte dans le calcul de la limite de simultanéité des tâches. AWS IoT Les tâches peuvent être déployées et annulées rapidement à un rythme de 1 000 appareils par minute. Chaque tâche n'est `concurrent` et ne compte dans le calcul de la limite de simultanéité des tâches que pendant une courte période. Une fois les exécutions des tâches déployées ou annulées, celles-ci ne sont plus simultanées et ne sont pas prises en compte dans le calcul de la limite de simultanéité des tâches. Vous pouvez utiliser la simultanéité des tâches pour créer un grand nombre de tâches en attendant que les appareils terminent l'exécution des tâches.

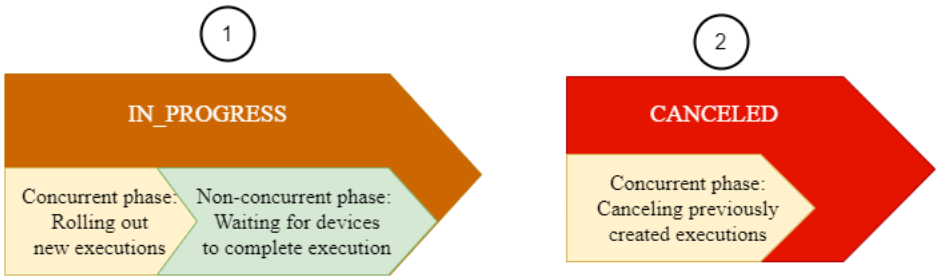
> **Note**
>
> Si une tâche avec la configuration de planification facultative et le déploiement du document de tâche prévu pour avoir lieu pendant une période de maintenance atteint la valeur sélectionnée `startTime` et que vous avez atteint votre limite maximale de simultanéité des tâches, cette tâche planifiée passera à un état de statut de. `CANCELED`

Pour déterminer si une tâche est simultanée, vous pouvez utiliser la `IsConcurrent` propriété d'une tâche depuis la AWS IoT console ou à l'aide de l'`ListJob`API `DescribeJob` or. Cette limite s'applique à la fois aux tâches continues et aux tâches instantanées.

Pour consulter les tâches actives, les limites de simultanéité des tâches et les autres quotas de AWS IoT tâches qui vous concernent Compte AWS et pour demander une augmentation de ces limites,

consultez la section Points de terminaison et quotas de gestion des AWS IoT appareils dans le. Références générales AWS

Le schéma suivant montre comment la simultanéité des tâches s'applique aux tâches en cours et aux tâches annulées.



> **Note**
>
> Les nouvelles tâches dont l'option est facultative `SchedulingConfig` conserveront un statut initial `SCHEDULED` et actualisera au niveau `IN_PROGRESS` une fois qu'elles auront atteint le niveau sélectionné `startTime`. Une fois que la nouvelle tâche facultative `SchedulingConfig` atteint la valeur sélectionnée `startTime` et actualisée au niveau `IN_PROGRESS`, elle est prise en compte dans le calcul de la limite des tâches actives et de la limite de simultanéité des tâches. Les tâches dont l'état de statut est égal à `SCHEDULED` seront prises en compte dans la limite des tâches actives, mais pas dans la limite de simultanéité des tâches.

Le tableau suivant indique les limites qui s'appliquent aux tâches actives et simultanées ainsi que les phases simultanées et non simultanées des états des tâches.

Limites de tâches actives et simultanées

Job status	Phase	Limite de tâches actives	Limite de simultanéité de la tâche
`SCHEDULED`	Phase non simultanée : AWS IoT Jobs attend le calendrier `startTime` de la tâche pour commencer à envoyer des notifications d'exécution à vos appareils.	S'applique	Ne s'applique pas

Job status	Phase	Limite de tâches actives	Limite de simultanéité de la tâche
	Les tâches de cette phase ne sont prises en compte que dans le calcul de la limite de tâches actives et leur `IsConcurrent` propriété sera définie sur faux.		
`IN_PROGRESS`	Phase simultanée : AWS IoT Jobs accepte la demande de création de la tâche et commence à envoyer des notifications d'exécution de la tâche sur vos appareils. Les tâches de cette phase sont simultanées, comme indiqué par la `IsConcurrent` propriété définie sur vrai, et sont prises en compte à la fois dans le calcul des tâches actives et des limites de simultanéité des tâches.	S'applique	S'applique
	Phase non simultanée : les AWS IoT tâches attendent que les appareils communiquent les résultats de leurs exécutions de tâches. Les tâches de cette phase ne sont prises en compte que dans le calcul de la limite de tâches actives et leur `IsConcurrent` propriété sera définie sur faux.	S'applique	Ne s'applique pas

Job status	Phase	Limite de tâches actives	Limite de simultanéité de la tâche
`Canceled`	Phase simultanée : AWS IoT Jobs accepte la demande d'annulation de la tâche et commence à annuler les exécutions de tâches créées précédemment pour vos appareils. Les tâches de cette phase sont simultanées et leur `IsConcurrent` propriété sera définie sur vrai. Une fois que la tâche et son exécution ont été annulées, la tâche n'est plus simultanée et n'est pas prise en compte dans le calcul de la limite de simultanéité des tâches.	Ne s'applique pas	S’applique

> **Note**
>
> La durée maximale d'une fenêtre de maintenance récurrente est de 23 heures et 50 minutes.

AWS IoT Device Management commandes

> **Important**
>
> Cette documentation décrit comment vous pouvez utiliser la fonctionnalité de commandes dans AWS IoT Device Management. Pour plus d'informations sur l'utilisation de cette fonctionnalité AWS IoT FleetWise, consultez la section Commandes à distance.
> Vous êtes seul responsable du déploiement des commandes d'une manière sûre et conforme aux lois applicables. Pour plus d'informations sur vos responsabilités, veuillez consulter les conditions AWS de service relatives AWS IoT aux services.

Utilisez AWS IoT Device Management des commandes pour envoyer une instruction depuis le cloud à un appareil connecté à AWS IoT. Les commandes ciblent un appareil à la fois et peuvent être utilisées pour des applications à faible latence et à haut débit, par exemple pour récupérer les journaux côté appareil ou pour initier un changement d'état de l'appareil.

La commande est une ressource réutilisable gérée par AWS IoT Device Management. Il contient des configurations qui sont appliquées avant d'être publiées sur l'appareil. Vous pouvez prédéfinir un ensemble de commandes pour des cas d'utilisation spécifiques, tels que l'allumage d'une ampoule ou le déverrouillage d'une portière de véhicule.

En utilisant la fonction de AWS IoT commandes, vous pouvez :

- Créez une ressource de commande et réutilisez sa configuration pour envoyer une commande plusieurs fois à votre appareil cible.
- Ciblez un appareil qui a été enregistré en tant qu' AWS IoT objet ou un client MQTT qui n'a pas été enregistré dans AWS IoT.
- Exécutez plusieurs commandes simultanément sur le périphérique cible sans le surcharger.
- Activez les notifications pour les événements liés aux commandes, puis récupérez et suivez l'état de l'appareil pendant qu'il exécute la commande jusqu'à sa fin.

Les rubriques suivantes expliquent comment créer des commandes, les envoyer à votre appareil et récupérer l'état signalé par l'appareil.

Rubriques

- Concepts et statut des commandes
- Flux de travail de commandes de haut niveau
- Création et gestion de commandes
- Démarrer et surveiller les exécutions de commandes
- Déprécier une ressource de commande

Concepts et statut des commandes

Utilisez AWS IoT des commandes pour envoyer une instruction depuis le cloud à un appareil connecté à AWS IoT. Pour utiliser la fonction de commandes :

1. Créez d'abord une ressource de commande avec une charge utile contenant les configurations requises pour exécuter la commande sur le périphérique.
2. Spécifiez l'équipement cible qui recevra la charge utile et exécutera les actions spécifiées.
3. Exécutez la commande sur le périphérique cible et récupérez les informations d'état sur le périphérique. Pour résoudre tout problème, consultez les CloudWatch journaux.

Pour de plus amples informations sur le contournement, veuillez consulter Flux de travail de commandes de haut niveau.

Rubriques

- Concepts clés des commandes
- États de commande
- État de l'exécution de la commande

Concepts clés des commandes

Voici quelques concepts clés relatifs à l'utilisation de la fonction de commandes.

Commandes

> Les commandes sont des instructions envoyées depuis le cloud à vos appareils IoT. Ces instructions (charge utile de commande) sont envoyées aux appareils sous forme de messages MQTT. Une fois que les appareils ont reçu la charge utile de la commande, ils peuvent traiter

les instructions pour effectuer l'action correspondante. Ces actions incluent par exemple la modification des paramètres de configuration de l'appareil, la transmission des relevés des capteurs ou le téléchargement de journaux. Les appareils peuvent ensuite exécuter la commande et renvoyer le résultat dans le cloud. Cela vous permet de surveiller et de contrôler à distance les appareils connectés.

Namespace

Lorsque vous utilisez la fonctionnalité des commandes, vous pouvez spécifier l'espace de noms de la commande. Lorsque vous souhaitez créer une commande dans AWS IoT Device Management, vous devez utiliser l'espace de `AWS-IoT` noms par défaut. Lorsque vous utilisez cet espace de noms, vous devez fournir une charge utile lors de la création de la commande. La charge utile sera utilisée lorsque vous exécuterez la commande sur votre appareil cible. Si vous souhaitez créer une commande pour à la AWS IoT FleetWise place, vous devez utiliser l'espace de `AWS-IoT-FleetWise` noms à la place. Pour plus d'informations, consultez la section Commandes distantes dans le guide du AWS IoT FleetWise développeur consacré aux commandes.

Charge utile

Lorsque vous créez la commande, vous devez fournir une charge utile qui définit les actions que le terminal doit effectuer. La charge utile peut utiliser le format de votre choix. Pour vous assurer que l'appareil peut lire et comprendre correctement les informations que vous envoyez, nous vous recommandons de spécifier le type de format de charge utile dans la commande. Si vos appareils l'utilisent MQTT5, ils peuvent suivre la norme MQTT pour identifier le format de charge utile. Un indicateur de format pour JSON ou CBOR sera disponible dans la rubrique de demande de commandes.

Appareil cible

Lorsque vous souhaitez exécuter la commande, vous devez spécifier une machine cible qui recevra la commande et exécutera les actions. Si votre appareil a été enregistré en tant qu'objet auprès de l'objet AWS IoT, vous pouvez utiliser le nom de l'objet. Si votre appareil n'a pas été enregistré, vous pouvez utiliser l'identifiant du client MQTT à la place. L'ID client est un identifiant unique pour votre appareil ou client défini dans le MQTT protocole. Il peut être utilisé pour connecter votre appareil à AWS IoT.

Exécution de commandes

Une exécution de commande est une instance d'une commande exécutée sur le périphérique cible. Lorsque vous lancez l'exécution, la commande (charge utile) est envoyée à l'équipement

cible. Un identifiant d'exécution de commande unique est désormais généré pour la cible. L'appareil peut ensuite exécuter la commande et rendre compte de sa progression à AWS IoT. La logique côté appareil détermine la manière dont la commande sera exécutée et la manière dont le statut sera publié dans les rubriques réservées.

Rubriques relatives aux commandes

Avant d'exécuter la commande, votre appareil doit être abonné à la rubrique de demande de commandes. Lorsque vous envoyez la demande au cloud pour exécuter la commande, la charge utile est envoyée à l'appareil dans la rubrique de demande de commande. Une fois que le périphérique a exécuté la commande, il peut publier le résultat et l'état de l'exécution dans la rubrique de réponse aux commandes. Pour de plus amples informations, veuillez consulter Rubriques relatives aux commandes.

États de commande

Une commande que vous créez dans votre compte Compte AWS peut être disponible, obsolète ou en attente de suppression.

Disponible

Une fois que vous avez créé avec succès une ressource de commande, elle sera disponible. La commande peut désormais être utilisée pour envoyer une exécution de commande à l'appareil.

Obsolète

Si vous n'avez plus l'intention d'utiliser une commande, vous pouvez la marquer comme obsolète. Dans cet état, vous ne pouvez pas envoyer de nouvelles exécutions de la commande sur vos appareils. Toutes les exécutions en attente qui ont déjà commencé continueront de s'exécuter sur l'appareil jusqu'à la fin. Pour envoyer de nouvelles exécutions, vous devez restaurer la commande afin qu'elle soit disponible.

Suppression en attente

Lorsque vous marquez une commande pour suppression, si la commande est devenue obsolète pendant une durée supérieure au délai maximum, elle est automatiquement supprimée. Cette action est permanente et ne peut être annulée. Par défaut, la durée maximale du délai d'expiration est de 12 heures. Si la commande n'est pas obsolète, ou si elle l'a été pendant une durée inférieure au délai maximum, elle sera en attente de suppression. La commande sera automatiquement supprimée de votre compte après la durée maximale du délai d'expiration.

État de l'exécution de la commande

Lorsque vous lancez l'exécution de la commande sur le périphérique cible, l'exécution de la commande entre dans un `CREATED` état. Il peut ensuite passer à l'un des autres états d'exécution des commandes en fonction de l'état signalé par le périphérique. Vous pouvez ensuite récupérer les informations d'état et suivre l'exécution de vos commandes.

> **Note**
>
> Pour un équipement cible donné, vous pouvez exécuter plusieurs commandes simultanément. Vous pouvez utiliser la fonction de contrôle de simultanéité pour limiter le nombre maximum d'exécutions envoyées au même appareil, afin d'éviter toute surcharge de celui-ci. Pour plus d'informations sur le nombre maximal d'exécutions simultanées que vous pouvez exécuter pour chaque appareil, consultez la section quotas de AWS IoT Device Management commandes.

Le tableau suivant montre les différents états d'exécution d'une commande et la manière dont l'exécution de la commande passe d'un statut à l'autre en fonction de la progression de l'exécution.

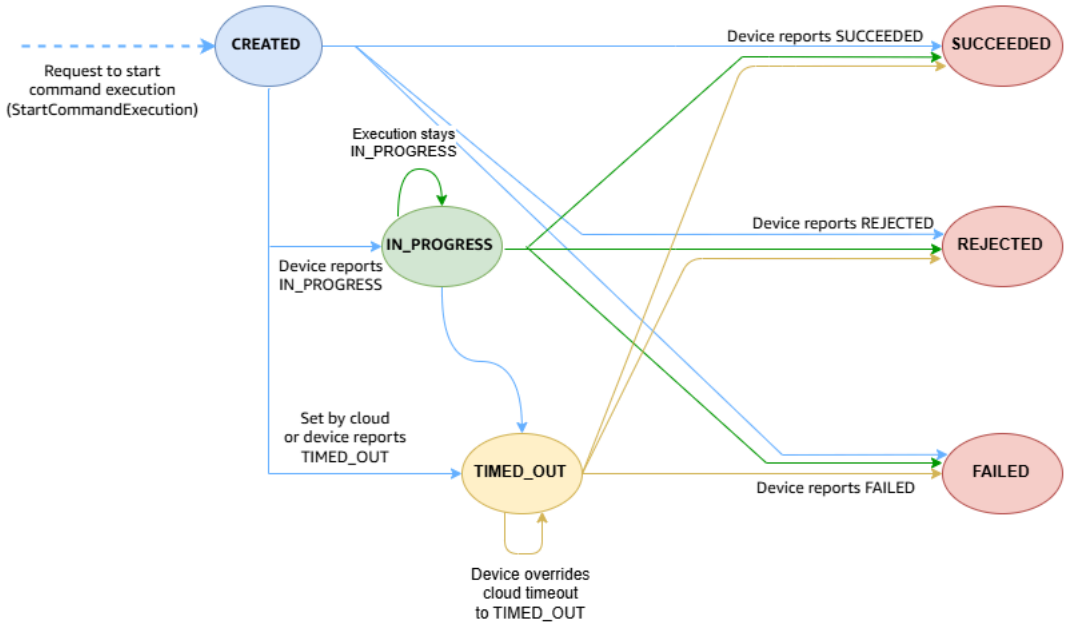
État et source de l'exécution des commandes

État de l'exécution de la commande	Initié par un appareil/le cloud ?	Exécution du terminal ?	Transitions de statut autorisées
`CREATED`	Cloud	Non	• EN_COURS • RÉUSSI • ÉCHEC • REFUSÉE • TIMED_OUT
`IN_PROGRESS`	Appareil	Non	• EN_COURS • RÉUSSI • ÉCHEC • REFUSÉE

État de l'exécution de la commande	Initié par un appareil/le cloud ?	Exécution du terminal ?	Transitions de statut autorisées
			• TIMED_OUT
`TIMED_OUT`	Appareil et cloud	Non	• RÉUSSI • ÉCHEC • REFUSÉE • TIMED_OUT
`SUCCEEDED`	Appareil	Oui	Ne s’applique pas
`FAILED`	Appareil	Oui	Ne s’applique pas
`REJECTED`	Appareil	Oui	Ne s’applique pas

Lorsque vos appareils exécutent la commande, ils peuvent publier des mises à jour de l'état et des résultats à tout moment sur le cloud en utilisant les sujets MQTT réservés aux commandes. Pour fournir un contexte supplémentaire sur l'état de chaque exécution de commande dans le cloud, celui-ci peut utiliser `reasonDescription` les `reasonCode` et contenus dans l'`statusReason`objet.

Le schéma suivant montre les différents états d'exécution des commandes et la manière dont la transition s'effectue entre eux.



> **Note**
>
> Lorsqu'aucune réponse de l'appareil n'est AWS IoT détectée dans le délai imparti, il est défini `TIMED_OUT` comme un statut temporaire qui autorise les nouvelles tentatives et les changements d'état. Si votre appareil effectue un signalement explicite `TIMED_OUT` lors de l'exécution de la commande, cela devient un état de terminal sans qu'aucune autre transition ne soit possible. Consultez Exécutions de commandes non terminales.

La section suivante décrit les exécutions de commandes terminales et non terminales, les différents statuts d'exécution et leur fonctionnement.

Rubriques

- Exécutions de commandes non terminales
- Exécutions des commandes du terminal

Exécutions de commandes non terminales

L'exécution de votre commande n'est pas terminale si elle peut accepter des mises à jour provenant d'appareils ou de clients. Une exécution dans un état non terminal est considérée comme active. Les statuts suivants ne sont pas terminaux.

- CRÉÉ

  Lorsque vous lancez l'exécution d'une commande depuis la AWS IoT console ou que vous utilisez l'`StartCommandExecution`API pour envoyer la commande à votre appareil à l'aide de la rubrique de demande de commandes. Si la demande aboutit, le statut d'exécution de la commande passe à`CREATED`. À partir de cet état, l'exécution de la commande peut passer à n'importe quel autre statut non terminal ou terminal.

- EN_COURS

  Après avoir reçu la charge utile de commande, votre appareil peut commencer à exécuter les instructions contenues dans la charge utile et effectuer les actions spécifiées. Pendant l'exécution de la commande, le dispositif peut publier une réponse au sujet de réponse aux commandes et mettre à jour l'état d'exécution de la commande en tant que`IN_PROGRESS`. À partir de l'`IN_PROGRESS`état, l'exécution de la commande peut passer à n'importe quel autre statut terminal ou non-terminal autre que. `CREATED`

  > **Note**
  >
  > L'`UpdateCommandExecution`API peut être invoquée plusieurs fois avec un statut de`IN_PROGRESS`. Vous pouvez spécifier des détails supplémentaires concernant l'exécution à l'aide de l'`statusReason`objet.

- TIMED_OUT

  Cet état d'exécution de commande peut être déclenché à la fois par le cloud et par l'appareil. Le `IN_PROGRESS` statut d'une exécution `CREATED` ou d'une exécution peut changer pour les raisons suivantes. `TIMED_OUT`

  - Une fois la commande envoyée à l'appareil, une minuterie démarre. S'il n'y a aucune réponse de l'appareil dans un délai spécifié, le cloud change l'état d'exécution de la commande en`TIMED_OUT`. Dans ce cas, l'exécution de la commande n'est pas terminale.
  - L'appareil peut remplacer l'état par n'importe quel autre état du terminal, ou signaler qu'un délai d'attente s'est produit lors de l'exécution de la commande, et définir le statut sur. `TIMED_OUT` Dans ce cas, le statut d'exécution reste `TIMED_OUT` fixe mais les champs de l'`StatusReason`objet changent en fonction des informations communiquées par les appareils. L'exécution de la commande devient alors un terminal.

  Pour de plus amples informations, veuillez consulter Valeur du délai d'expiration et statut `TIMED_OUT` d'exécution.

Exécutions des commandes du terminal

Une exécution de commande devient terminale si l'exécution n'accepte plus de mises à jour supplémentaires de la part des appareils. Les statuts suivants sont terminaux. Une exécution peut passer aux états du terminal à partir de n'importe quel statut non terminal,`CREATED`, `IN_PROGRESS` ou. `TIMED_OUT`

- RÉUSSI

  Si le périphérique a correctement exécuté la commande, il peut publier une réponse à la rubrique de réponse aux commandes et mettre à jour l'état d'exécution de la commande sur`SUCCEEDED`.

- ÉCHEC

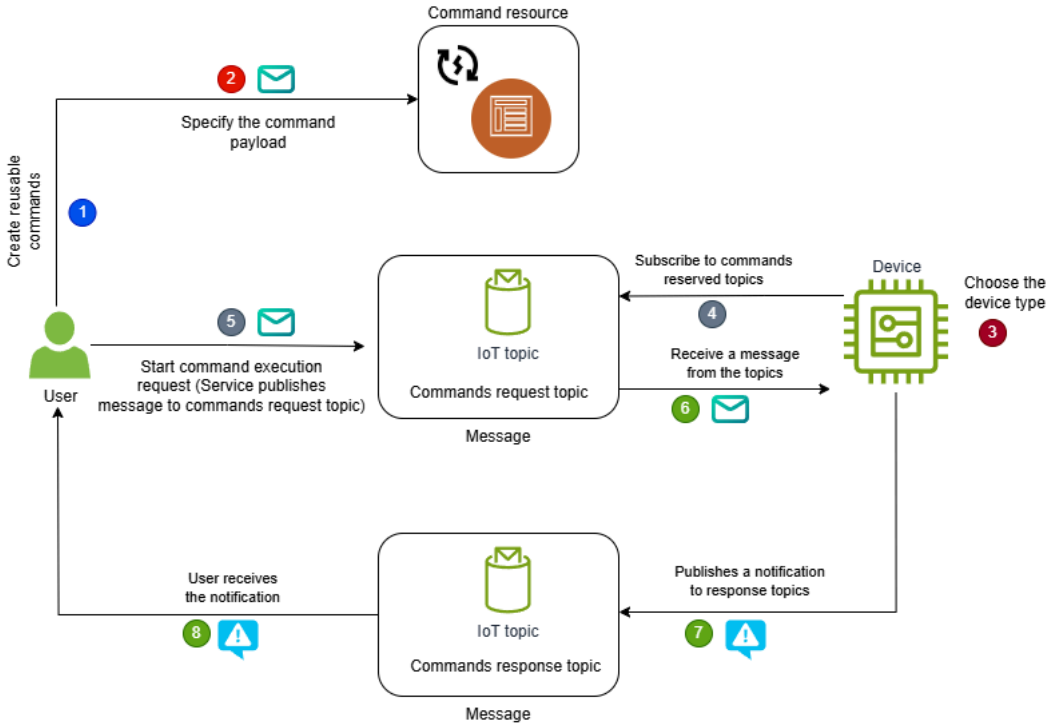
  Lorsque votre appareil ne parvient pas à exécuter la commande, il peut publier une réponse à la rubrique de réponse aux commandes et mettre à jour l'état d'exécution de la commande sur`FAILED`. Vous pouvez utiliser les `reasonDescription` champs `reasonCode` et de l'`statusReason`objet, ou les CloudWatch journaux, pour résoudre les problèmes de manière plus approfondie.

- REFUSÉE

  Lorsque votre appareil reçoit une demande non valide ou incompatible, il peut appeler l'`UpdateCommandExecution`API avec un statut de`REJECTED`. Vous pouvez utiliser les `reasonDescription` champs `reasonCode` et de l'`statusReason`objet, ou les CloudWatch journaux, pour résoudre les problèmes éventuels.

Flux de travail de commandes de haut niveau

Les étapes suivantes fournissent une vue d'ensemble du flux de travail des commandes entre vos appareils et vos AWS IoT Device Management commandes. Lorsque vous utilisez l'une des commandes des opérations de l'API HTTP, la demande est signée à l'aide des informations d'identification Sigv4.



Présentation du flux de travail

- Création et gestion de commandes
- Choisissez l'appareil cible pour vos commandes et abonnez-vous aux rubriques MQTT
- Démarrez et surveillez les exécutions de commandes pour votre appareil cible
- (Facultatif) Activer les notifications pour les événements liés aux commandes

Création et gestion de commandes

Pour créer et gérer des commandes pour vos appareils, effectuez les étapes suivantes.

1. Création d'une ressource de commande

   Avant de pouvoir envoyer la commande à vos appareils, créez une ressource de commande à partir du hub de commande de la AWS IoT console ou à l'aide de l'API du plan de `CreateCommand`contrôle.

2. Spécifiez la charge utile

   Lors de la création de la commande, vous devez fournir une charge utile pour votre commande. Le contenu de la charge utile peut utiliser n'importe quel format de votre choix. Pour vous

assurer que l'appareil interprète correctement la charge utile, nous vous recommandons de spécifier également le type de contenu de la charge utile.

3. (Facultatif) Gérez les commandes créées

   Après avoir créé la commande, vous pouvez mettre à jour son nom d'affichage et sa description. Vous pouvez également marquer une commande comme obsolète si vous n'avez plus l'intention de l'utiliser, ou la supprimer complètement de votre compte. Si vous souhaitez modifier les informations de charge utile, vous devez créer une nouvelle commande et télécharger le nouveau fichier de charge utile.

Choisissez l'appareil cible pour vos commandes et abonnez-vous aux rubriques MQTT

Pour préparer le flux de travail des commandes, choisissez votre appareil cible et spécifiez les sujets MQTT AWS IoT réservés pour recevoir des commandes et publier des messages de réponse.

1. Choisissez l'appareil cible pour votre commande

   Pour préparer le flux de travail des commandes, choisissez votre appareil cible qui recevra la commande et exécutera les actions spécifiées. L'appareil cible peut être un AWS IoT objet que vous avez enregistré dans le AWS IoT registre, ou peut être spécifié à l'aide de l'ID client MQTT, si votre appareil n'a pas été enregistré auprès AWS IoT de celui-ci. Pour de plus amples informations, veuillez consulter Considérations relatives à l'appareil cible.

2. Configuration de la politique relative aux appareils IoT

   Avant que votre appareil puisse recevoir des exécutions de commandes et publier des mises à jour, il doit utiliser une politique IAM qui accorde les autorisations nécessaires pour effectuer ces actions. Pour des exemples de politiques que vous pouvez utiliser selon que votre appareil est enregistré en tant qu' AWS IoT objet ou qu'il est spécifié sous forme d'ID client MQTT, consultezExemple de politique IAM.

3. Établir une connexion MQTT

   Pour préparer vos appareils à utiliser la fonction de commandes, ils doivent d'abord se connecter au courtier de messages et s'abonner aux rubriques de demande et de réponse. Votre appareil doit être autorisé à effectuer l'`iot:Connect`action de connexion AWS IoT Core et d'établissement d'une connexion MQTT avec le courtier de messages. Pour trouver le point de terminaison du plan de données qui vous convient Compte AWS, utilisez

l'`DescribeEndpoint`API ou la commande `describe-endpoint` CLI, comme indiqué ci-dessous.

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

L'exécution de cette commande renvoie le point de terminaison du plan de données spécifique au compte, comme indiqué ci-dessous.

```
account-specific-prefix.iot.region.amazonaws.com
```

4. Abonnez-vous aux rubriques relatives aux commandes

Une fois la connexion établie, vos appareils peuvent s'abonner à la rubrique de demande de commandes. Lorsque vous créez une commande et que vous lancez l'exécution de la commande sur votre appareil cible, le message de charge utile est publié dans le sujet de la demande par le courtier de messages. Votre appareil peut ensuite recevoir le message de charge utile et traiter la commande.

(Facultatif) Vos appareils peuvent également s'abonner à ces sujets de réponse aux commandes (`acceptedourejected`) pour recevoir un message indiquant si le service cloud a accepté ou rejeté la réponse de l'appareil.

Dans cet exemple, remplacez :

- *<device>*avec `thing` ou `client` selon que l'appareil que vous ciblez a été enregistré en tant qu'objet IoT ou spécifié en tant que client MQTT.
- *<DeviceID>*avec l'identifiant unique de votre appareil cible. Cet identifiant peut être l'identifiant unique du client MQTT ou un nom d'objet.

> **Note**
>
> Si le type de charge utile n'est pas JSON ou CBOR, le *<PayloadFormat>* champ n'est peut-être pas présent dans la rubrique de demande de commandes. Pour obtenir le format de charge utile, nous vous recommandons d'utiliser MQTT 5 pour obtenir les informations de format à partir des en-têtes de message MQTT. Pour de plus amples informations, veuillez consulter Rubriques relatives aux commandes.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/accepted/<PayloadFormat>
$aws/commands/<devices>/<DeviceID>/executions/+/response/rejected/<PayloadFormat>
```

Démarrez et surveillez les exécutions de commandes pour votre appareil cible

Après avoir créé les commandes et spécifié les cibles de la commande, vous pouvez démarrer l'exécution sur le périphérique cible en effectuant les étapes suivantes.

1. Lancer l'exécution de la commande sur le périphérique cible

   Démarrez l'exécution de la commande sur l'appareil cible à partir du hub de commande de la AWS IoT console ou à l'aide de l'API du plan de `StartCommandExecution` données avec le point de terminaison spécifique à votre compte`iot:Jobs`. L'API publie le message de charge utile dans la rubrique de demande de commandes mentionnée ci-dessus à laquelle l'appareil s'est abonné.

   > **Note**
   >
   > Si l'appareil était hors ligne lorsque la commande a été envoyée depuis le cloud et s'il utilise des sessions persistantes MQTT, la commande attend le courtier de messages. Si l'appareil revient en ligne avant la fin du délai imparti et s'il s'est abonné à la rubrique de demande de commandes, il peut alors traiter la commande et publier le résultat dans la rubrique de réponse aux commandes. Si l'appareil ne revient pas en ligne avant le délai d'expiration, l'exécution de la commande expirera et le message de charge utile risque d'expirer et d'être supprimé par le courtier de messages.

2. Mettre à jour le résultat de l'exécution de la commande

   L'appareil reçoit désormais le message de charge utile et peut traiter la commande et exécuter les actions spécifiées, puis publier le résultat de l'exécution de la commande dans la rubrique de réponse aux commandes suivante à l'aide de l'`UpdateCommandExecution`API. Si votre appareil s'est abonné aux sujets de réponse acceptés et rejetés des commandes, il recevra un message indiquant si la réponse a été acceptée ou rejetée par le service cloud.

Selon la manière que vous avez spécifiée dans le sujet de la demande, il *<devices>* peut s'agir d'objets ou de clients, et il *<DeviceID>* peut s'agir du nom de votre objet IoT ou de l'ID client MQTT.

> **Note**
>
> Il ne *<PayloadFormat>* peut s'agir que de JSON ou CBOR dans la rubrique de réponse aux commandes.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/
response/<PayloadFormat>
```

3. (Facultatif) Récupère le résultat de l'exécution de la commande

   Pour récupérer le résultat de l'exécution de la commande, vous pouvez consulter l'historique des commandes depuis la AWS IoT console ou utiliser l'opération API du plan de `GetCommandExecution` contrôle. Pour obtenir les informations les plus récentes, votre appareil doit avoir publié le résultat de l'exécution de la commande dans la rubrique consacrée aux réponses aux commandes. Vous pouvez également obtenir des informations supplémentaires sur les données d'exécution, telles que la date de leur dernière mise à jour, le résultat de l'exécution et la date à laquelle l'exécution s'est terminée.

(Facultatif) Activer les notifications pour les événements liés aux commandes

Vous pouvez vous abonner aux événements de commandes pour recevoir des notifications lorsque le statut de l'exécution d'une commande change. Les étapes suivantes vous montrent comment vous abonner à des événements de commandes, puis comment les traiter.

1. Créer une règle de rubrique

   Vous pouvez vous abonner à la rubrique des événements relatifs aux commandes et recevoir des notifications lorsque le statut de l'exécution d'une commande change. Vous pouvez également créer une règle thématique pour acheminer les données traitées par l'appareil vers d'autres AWS IoT services pris en charge par des règles AWS Lambda, tels qu'Amazon SQS et AWS Step Functions. Vous pouvez créer une règle de sujet à l'aide de la AWS IoT console ou

à l'aide de l'API du plan de `CreateTopicRule` AWS IoT Core contrôle. Pour de plus amples informations, veuillez consulter Création d'une AWS IoT règle.

Dans cet exemple, remplacez-le *<CommandID>* par l'identifiant de la commande pour laquelle vous souhaitez recevoir des notifications et *<CommandExecutionStatus>* par le statut de l'exécution de la commande.

```
$aws/events/commandExecution/<CommandID>/<CommandExecutionStatus>
```

> **Note**
>
> Pour recevoir des notifications concernant toutes les commandes et les statuts d'exécution des commandes, vous pouvez utiliser des caractères génériques et vous abonner à la rubrique suivante.

```
$aws/events/commandExecution/+/#
```

2. Réception et traitement des événements liés aux commandes

   Si vous avez créé une règle thématique à l'étape précédente pour vous abonner aux événements de commandes, vous pouvez gérer les notifications push de commandes que vous recevez et créer une application sur la base de ces services.

Le code suivant montre un exemple de charge utile pour les notifications d'événements de commande que vous recevrez.

```
{
    "executionId": "2bd65c51-4cfd-49e4-9310-d5cbfdbc8554",
    "status":"FAILED",
    "statusReason": {
         "reasonCode": "DEVICE_TOO_BUSY",
         "reasonDescription": ""
    },
    "eventType": "COMMAND_EXECUTION",
    "commandArn":"arn:aws:iot:us-east-1:123456789012:command/0b9d9ddf-
e873-43a9-8e2c-9fe004a90086",
    "targetArn":"arn:aws:iot:us-east-1:123456789012:thing/5006c3fc-
de96-4def-8427-7eee36c6f2bd",
```

```
    "timestamp":1717708862107
}
```

Création et gestion de commandes

Vous pouvez utiliser la fonction de AWS IoT Device Management commandes pour configurer des actions à distance réutilisables ou pour envoyer des instructions immédiates et ponctuelles à vos appareils. Les sections suivantes expliquent comment créer et gérer des commandes à partir de la AWS IoT console et à l'aide du AWS CLI.

Création et gestion des opérations de commande

- Création d'une ressource de commande
- Récupérer des informations sur une commande
- Répertoriez les commandes dans votre Compte AWS
- Mettre à jour une ressource de commande
- Dépréciation ou restauration d'une ressource de commande
- Supprimer une ressource de commande

Création d'une ressource de commande

Lorsque vous créez une commande, vous devez fournir les informations suivantes.

- Informations générales

  Lorsque vous créez une commande, vous devez fournir un ID de commande, qui est un identifiant unique pour vous aider à identifier la commande lorsque vous souhaitez l'exécuter sur le périphérique cible. Vous pouvez également éventuellement spécifier un nom d'affichage, une description et des balises pour mieux gérer la commande.

- Charge utile

  Vous devez également fournir une charge utile qui définit les actions que l'appareil doit effectuer. Bien que facultatif, nous vous recommandons de spécifier le type de format de charge utile afin que le périphérique interprète correctement la charge utile.

Sujets relatifs à la charge utile et aux commandes

Les rubriques réservées aux commandes utilisent un format qui dépend du type de format de charge utile.

- Si vous spécifiez un type de contenu de charge utile de `application/json` ou`application/cbor`, le sujet de la demande sera le suivant.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

- Si vous spécifiez un type de contenu de charge utile autre que `application/json` ou`application/cbor`, ou si vous ne spécifiez pas le type de format de charge utile, le sujet de la demande sera le suivant. Dans ce cas, le format de charge utile sera inclus dans l'en-tête du message MQTT.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

La rubrique de réponse aux commandes renverra un format utilisant `json` ou `cbor` indépendant du type de format de charge utile. Le sujet de réponse utilisera le format suivant où *<PayloadFormat>* doit être `json` ou`cbor`.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Création d'une ressource de commande (console)

Les sections suivantes présentent les considérations relatives au format de charge utile des commandes et expliquent comment créer des commandes à partir de la console.

Rubriques

- Format de charge utile de commande
- Comment créer une commande (console)

Format de charge utile de commande

La charge utile peut utiliser le format de votre choix. La taille maximale de la charge utile ne doit pas dépasser 32 Ko. Pour garantir que le périphérique peut interpréter correctement et en toute sécurité la charge utile, nous vous recommandons de spécifier le type de format de charge utile.

Vous spécifiez le type de format de charge utile à l'aide du `type/subtype` format, tel que `application/json` ou`application/cbor`. Par défaut, il sera défini comme`application/octet-stream`. Pour plus d'informations sur les formats de charge utile que vous pouvez spécifier, consultez la section Types MIME courants.

Comment créer une commande (console)

Pour créer une commande à partir de la console, accédez au hub de commande de la AWS IoT console et effectuez les étapes suivantes.

1. Pour créer une nouvelle ressource de commande, choisissez Créer une commande.
2. Spécifiez un ID de commande unique pour vous aider à identifier la commande que vous souhaitez exécuter sur le périphérique cible.
3. (Facultatif) Spécifiez un nom d'affichage facultatif, une description et toute paire nom-valeur en tant que balises pour votre commande.
4. Téléchargez le fichier de charge utile depuis votre stockage local qui contient les actions que l'appareil doit effectuer. Bien que facultatif, nous vous recommandons de spécifier le type de format de charge utile afin que le périphérique interprète correctement le fichier et traite les instructions.
5. Choisissez Créer une commande.

Création d'une ressource de commande (CLI)

Cette section décrit le fonctionnement de l'API du plan de contrôle HTTP et la AWS CLI commande correspondante `create-command`que vous pouvez exécuter pour créer une ressource de commande. `CreateCommand`

Rubriques

- Charge utile de commande
- Exemple de politique IAM
- Exemple de création de commande

Charge utile de commande

Lors de la création de la commande, vous devez fournir une charge utile. La charge utile que vous fournissez est codée en base64. Lorsque vos appareils reçoivent la commande, la logique côté

périphérique peut traiter la charge utile et effectuer les actions spécifiées. Pour vous assurer que vos appareils reçoivent correctement la commande et la charge utile, nous vous recommandons de spécifier le type de contenu de la charge utile.

> **Note**
>
> Après avoir créé la commande, vous ne pouvez pas modifier la charge utile. Pour modifier la charge utile, vous devez créer une nouvelle commande.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`CreateCommand`action.

Dans cet exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple*`ap-south-1`*.
- *`account-id`*avec votre Compte AWS numéro, par exemple*`123456789012`*.
- *`command-id`*avec un identifiant unique pour votre identifiant de AWS IoT commande, tel que*`LockDoor`*. Si vous souhaitez envoyer plusieurs commandes, vous pouvez les spécifier dans la section Ressource de la stratégie IAM.

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": "iot:CreateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:<region>:<account_id>:command/command-id"

    }
}
```

Exemple de création de commande

L'exemple suivant montre comment créer une commande. En fonction de votre application, remplacez :

- *<command-id>*avec un identifiant unique pour la commande. Par exemple, pour verrouiller l'historique documentaire de votre maison, vous pouvez spécifier*LockDoor*. Nous vous recommandons d'utiliser l'UUID. Vous pouvez également utiliser des caractères alphanumériques, « - » et « _ ».
- (Facultatif) *<display-name>* et *<description>* qui sont des champs facultatifs que vous pouvez utiliser pour fournir un nom convivial et une description significative à la commande, tels que*Lock the doors of my home*.
- `namespace`, que vous pouvez utiliser pour spécifier l'espace de noms de la commande. Ça doit l'êtreAWS-IoT.
- `payload`contient des informations sur la charge utile que vous souhaitez utiliser lors de l'exécution de la commande et son type de contenu.

```
aws iot create-command \
    --command-id <command-id> \
    --display-name <display-name> \
    --description <description> \
    --namespace AWS-IoT \
    --payload
 '{"content":"eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=","contentType":"application/json"}'
```

L'exécution de cette commande génère une réponse contenant l'ID et l'ARN (nom de ressource Amazon) de la commande. Par exemple, si vous avez spécifié la *LockDoor* commande lors de sa création, voici un exemple de sortie d'exécution de la commande.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:ap-south-1:123456789012:command/LockDoor"
}
```

Récupérer des informations sur une commande

Après avoir créé une commande, vous pouvez récupérer des informations la concernant depuis la AWS IoT console et en utilisant le AWS CLI. Vous pouvez obtenir les informations suivantes.

- L'ID de commande, le nom de la ressource Amazon (ARN), tout nom d'affichage et toute description que vous avez spécifiés pour la commande.
- L'état de la commande, qui indique si une commande est disponible pour être exécutée sur le périphérique cible, ou si elle est obsolète ou supprimée.
- La charge utile que vous avez fournie et son type de format.
- Heure à laquelle la commande a été créée et mise à jour pour la dernière fois.

Récupérer une ressource de commande (console)

Pour récupérer une commande depuis la console, accédez au hub de commande de la AWS IoT console, puis choisissez la commande que vous avez créée pour en afficher les détails.

Outre les détails de la commande, vous pouvez consulter l'historique des commandes, qui fournit des informations sur l'exécution de la commande sur le périphérique cible. Après avoir exécuté cette commande sur l'appareil, vous trouverez des informations sur les exécutions dans cet onglet.

Récupérer une ressource de commande (CLI)

Utilisez l'opération API du plan de contrôle `GetCommand`HTTP ou la `get-command` AWS CLI commande pour récupérer des informations sur une ressource de commande. Vous devez déjà avoir créé la commande à l'aide de la demande `CreateCommand` d'API ou de la `create-command` CLI.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`GetCommand`action.

Dans cet exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`ap-south-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple*`123456789023`*.
- *`command-id`*avec votre identifiant de commande AWS IoT unique, tel que*`LockDoor`*. Si vous souhaitez récupérer plusieurs commandes, vous pouvez les spécifier dans la section Ressource de la politique IAM.

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": "iot:GetCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:<region>:<account_id>:command/command-id"

    }
}
```

Récupérez un exemple de commande (AWS CLI)

L'exemple suivant montre comment récupérer des informations sur une commande à l'aide du `get-command` AWS CLI. En fonction de votre application, *<command-id>* remplacez-le par l'identifiant de la commande pour laquelle vous souhaitez récupérer des informations. Vous pouvez obtenir ces informations à partir de la réponse de la `create-command` CLI.

```
aws iot get-command --command-id <command-id>
```

L'exécution de cette commande génère une réponse contenant des informations sur la commande, la charge utile, ainsi que l'heure à laquelle elle a été créée et mise à jour pour la dernière fois. Il fournit également des informations indiquant si une commande est obsolète ou en cours de suppression.

Par exemple, le code suivant montre un exemple de réponse.

```
{
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:<region>:<account>:command/LockDoor",
    "namespace": "AWS-IoT",
    "payload":{
        "content": "eyAibWVzc2FnZSI6ICJIZWxsbyBJb1QiIH0=",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-03-23T00:50:10.095000-07:00",
    "deprecated": false,
```

```
    "pendingDeletion": false
}
```

Répertoriez les commandes dans votre Compte AWS

Après avoir créé des commandes, vous pouvez consulter celles que vous avez créées dans votre compte. Dans la liste, vous trouverez des informations sur :

- L'ID de commande et tout nom d'affichage que vous avez spécifié pour les commandes.
- Le nom de ressource Amazon (ARN) des commandes.
- État de la commande qui indique si les commandes peuvent être exécutées sur le périphérique cible ou si elles sont obsolètes.

> **Note**
>
> La liste des personnes en cours de suppression de votre compte ne s'affiche pas. Si les commandes sont en attente de suppression, vous pouvez toujours consulter les détails de ces commandes à l'aide de leur ID de commande.

- Heure à laquelle les commandes ont été créées et mises à jour pour la dernière fois.

Répertorier les commandes de votre compte (console)

Dans la AWS IoT console, vous pouvez trouver la liste des commandes que vous avez créées et leurs détails en accédant au Command Hub.

Répertorier les commandes de votre compte (CLI)

Pour répertorier les commandes que vous avez créées, utilisez l'opération `ListCommands`API ou la `list-commands`CLI.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`ListCommands`action.

Dans cet exemple, remplacez :

- *region*avec votre Région AWS, par exemple`ap-south-1`.
- *account-id*avec votre Compte AWS numéro, par exemple*123456789012*.

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": "iot:ListCommands",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:<region>:<account_id>:command/*"
    }
}
```

Exemple de liste des commandes dans votre compte

La commande suivante indique comment répertorier les commandes de votre compte.

```
aws iot list-commands --namespace "AWS-IoT"
```

L'exécution de cette commande génère une réponse contenant une liste des commandes que vous avez créées, l'heure à laquelle les commandes ont été créées et la date de leur dernière mise à jour. Il fournit également des informations sur l'état de la commande, qui indiquent si une commande est obsolète ou si elle est disponible pour être exécutée sur le périphérique cible. Pour plus d'informations sur les différents statuts et la raison du statut, consultezÉtat de l'exécution de la commande.

Mettre à jour une ressource de commande

Après avoir créé une commande, vous pouvez mettre à jour le nom d'affichage et la description de la commande.

> **Note**
>
> La charge utile de la commande ne peut pas être mise à jour. Pour mettre à jour ces informations ou utiliser une charge utile modifiée, vous devez créer une nouvelle commande.

Mettre à jour une ressource de commande (console)

Pour mettre à jour une commande depuis la console, accédez au Command Hub de la AWS IoT console et effectuez les étapes suivantes.

1. Pour mettre à jour une ressource de commande existante, choisissez la commande que vous souhaitez mettre à jour, puis sous Actions, choisissez Modifier.
2. Spécifiez le nom d'affichage et la description que vous souhaitez utiliser, ainsi que toutes les paires nom-valeur en tant que balises pour votre commande.
3. Choisissez Modifier pour enregistrer la commande avec les nouveaux paramètres.

Mettre à jour une ressource de commande (CLI)

Utilisez l'opération API du plan de `UpdateCommand`contrôle ou le `update-command` AWS CLI pour mettre à jour une ressource de commande. Grâce à cette API, vous pouvez :

- Modifiez le nom d'affichage et la description d'une commande que vous avez créée.
- Dépréciez une ressource de commande ou restaurez une commande déjà obsolète.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`UpdateCommand`action.

Dans cet exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`ap-south-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple*`123456789012`*.
- *`command-id`*avec votre identifiant de commande AWS IoT unique, tel que*`LockDoor`*. Si vous souhaitez récupérer plusieurs commandes, vous pouvez les spécifier dans la section Ressource de la politique IAM.

JSON

```
{
```

```
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": "iot:UpdateCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:<region>:<account_id>:command/<command-id>"

    }
}
```

Exemples de mise à jour des informations relatives à une commande (AWS CLI)

L'exemple suivant montre comment mettre à jour les informations relatives à une commande à l'aide de cette `update-command` AWS CLI commande. Pour plus d'informations sur la façon dont vous pouvez utiliser cette API pour déprécier ou restaurer une ressource de commande, consultez. Mettre à jour une ressource de commande (CLI)

L'exemple montre comment mettre à jour le nom d'affichage et la description d'une commande. En fonction de votre application, *<command-id>* remplacez-le par l'identifiant de la commande pour laquelle vous souhaitez récupérer des informations.

```
aws iot update-command \
    --command-id <command-id>
    --displayname <display-name> \
    --description <description>
```

L'exécution de cette commande génère une réponse contenant les informations mises à jour sur la commande et l'heure de sa dernière mise à jour. Le code suivant montre un exemple de demande et de réponse pour mettre à jour le nom d'affichage et la description d'une commande qui met le courant alternatif hors tension.

```
aws iot update-command \
    --command-id <LockDoor> \
    --displayname <Secondary lock door> \
    --description <Locks doors to my home>
```

L'exécution de cette commande génère la réponse suivante.

```
{
```

```
    "commandId": "LockDoor",
    "commandArn": "arn:aws:iot:ap-south-1:123456789012:command/LockDoor",
    "displayName": "Secondary lock door",
    "description": "Locks doors to my home",
    "lastUpdatedAt": "2024-05-09T23:15:53.899000-07:00"
}
```

Dépréciation ou restauration d'une ressource de commande

Après avoir créé une commande, si vous ne souhaitez plus continuer à l'utiliser, vous pouvez la marquer comme obsolète. Lorsque vous désapprouvez une commande, toutes les exécutions de commandes en attente continuent de s'exécuter sur le périphérique cible jusqu'à ce qu'elles atteignent le statut de terminal. Une fois qu'une commande est devenue obsolète, si vous souhaitez l'utiliser, par exemple pour envoyer une nouvelle exécution de commande à la machine cible, vous devez la restaurer.

> **Note**
>
> Vous ne pouvez pas modifier une commande obsolète, ni exécuter de nouvelles exécutions pour celle-ci. Pour exécuter de nouvelles commandes sur l'appareil, vous devez le restaurer afin que l'état de la commande passe à Disponible.

Pour plus d'informations sur la dépréciation et la restauration d'une commande, ainsi que sur les considérations associées, consultez. Déprécier une ressource de commande

Supprimer une ressource de commande

Si vous ne souhaitez plus utiliser une commande, vous pouvez la supprimer définitivement de votre compte. Si l'action de suppression est réussie :

- Si la commande est devenue obsolète pendant une durée supérieure au délai maximum de 12 heures, elle sera immédiatement supprimée.
- Si la commande n'est pas obsolète, ou si elle l'a été pendant une durée inférieure au délai maximum, la commande sera dans un état. `pending deletion` Il sera automatiquement supprimé de votre compte après le délai maximum de 12 heures.

> **Note**
>
> La commande peut être supprimée même si des exécutions de commandes sont en attente. La commande sera en attente de suppression et sera automatiquement supprimée de votre compte.

Supprimer une ressource de commande (console)

Pour supprimer une commande de la console, accédez au Command Hub de la AWS IoT console et effectuez les étapes suivantes.

1. Choisissez la commande que vous souhaitez supprimer, puis sous Actions, choisissez Supprimer.
2. Confirmez que vous souhaitez supprimer la commande, puis choisissez Supprimer.

La commande sera marquée pour suppression et sera définitivement supprimée de votre compte au bout de 12 heures.

Supprimer une ressource de commande (CLI)

Utilisez l'opération d'API du plan de contrôle `DeleteCommand` HTTP ou la `delete-command` AWS CLI commande pour supprimer une ressource de commande. Si l'action de suppression aboutit, vous verrez un HTTP `statusCode` 204 ou 202, et la commande sera automatiquement supprimée de votre compte après le délai d'expiration maximal de 12 heures. Dans le cas du statut 204, cela indique que la commande a été supprimée.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`DeleteCommand`action.

Dans cet exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`ap-south-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple*`123456789012`*.

- *command-id*avec votre identifiant de commande AWS IoT unique, tel que*LockDoor*. Si vous souhaitez récupérer plusieurs commandes, vous pouvez les spécifier dans la section Ressource de la politique IAM.

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": "iot:DeleteCommand",
        "Effect": "Allow",
        "Resource": "arn:aws:iot:<region>:<account_id>:command/command-id"

    }
}
```

Exemple de suppression d'une commande (AWS CLI)

Les exemples suivants montrent comment supprimer une commande à l'aide de cette `delete-command` AWS CLI commande. En fonction de votre application, *<command-id>* remplacez-le par l'identifiant de la commande que vous supprimez.

```
aws iot delete-command --command-id <command-id>
```

Si la demande d'API aboutit, la commande génère un code d'état 202 ou 204. Vous pouvez utiliser l'`GetCommand`API pour vérifier que la commande n'existe plus dans votre compte.

Démarrer et surveiller les exécutions de commandes

Après avoir créé une ressource de commande, vous pouvez démarrer une exécution de commande sur le périphérique cible. Une fois que le périphérique commence à exécuter la commande, il peut commencer à mettre à jour le résultat de l'exécution de la commande et publier des mises à jour de statut et des informations sur les résultats dans les rubriques réservées au MQTT. Vous pouvez ensuite récupérer le statut de l'exécution de la commande et surveiller le statut des exécutions dans votre compte.

Cette section explique comment démarrer et contrôler des commandes à l'aide de la AWS IoT console et du AWS CLI.

Démarrer et surveiller les opérations liées aux commandes

- Lancer l'exécution d'une commande
- Mettre à jour le résultat de l'exécution d'une commande
- Récupérer l'exécution d'une commande
- Affichage des mises à jour des commandes à l'aide du client de test MQTT
- Répertoriez les exécutions de commandes dans votre Compte AWS
- Supprimer l'exécution d'une commande

Lancer l'exécution d'une commande

> **Important**
>
> Vous êtes seul responsable du déploiement des commandes d'une manière sûre et conforme aux lois applicables.

Avant de lancer l'exécution d'une commande, vous devez vous assurer que :

- Vous avez créé une commande dans l'espace de AWS IoT noms et fourni les informations de charge utile. Lorsque vous commencez à exécuter la commande, le périphérique traite les instructions contenues dans la charge utile et exécute les actions spécifiées. Pour plus d'informations sur la création de commandes, consultezCréation d'une ressource de commande.
- Votre appareil s'est abonné aux rubriques réservées aux commandes du MQTT. Lorsque vous lancez l'exécution de la commande, les informations de charge utile sont publiées dans la rubrique réservée de demande MQTT suivante.

  Dans ce cas, il *<devices>* peut s'agir d'objets IoT ou de clients MQTT, et *<DeviceID>* il s'agit du nom de l'objet ou de l'ID du client. Les fichiers pris en charge *<PayloadFormat>* sont JSON et CBOR. Pour plus d'informations sur les sujets relatifs aux commandes, consultezRubriques relatives aux commandes.

  ```
  $aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
  ```

S'il ne s'*<PayloadFormat>*agit pas de JSON et de CBOR, le format de rubrique des commandes ci-dessous est illustré.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

Considérations relatives à l'appareil cible

Lorsque vous souhaitez exécuter la commande, vous devez spécifier la machine cible qui recevra la commande et exécutera les instructions spécifiées. L'appareil cible peut être soit un AWS IoT objet, soit l'ID du client si le périphérique n'a pas été enregistré dans le AWS IoT registre. Après avoir reçu la charge utile de la commande, le périphérique peut commencer à exécuter la commande et effectuer les actions spécifiées.

AWS IoT chose

Le périphérique cible de la commande peut être un AWS IoT objet que vous avez enregistré dans le AWS IoT registre des objets. Les fonctionnalités AWS IoT facilitent la recherche et la gestion de vos appareils.

Vous pouvez enregistrer votre appareil en tant qu'objet lorsque vous le connectez à AWS IoT partir de la page Connect device ou à l'aide de l'`CreateThing`API. Vous pouvez trouver un objet existant pour lequel vous souhaitez exécuter la commande depuis la page Thing Hub de la AWS IoT console ou à l'aide de l'`DescribeThing`API. Pour plus d'informations sur la façon d'enregistrer votre appareil en tant qu' AWS IoT objet, consultez la section Gestion des éléments avec le registre.

ID de client

Si votre appareil n'a pas été enregistré auprès d'un appareil AWS IoT, vous pouvez utiliser l'identifiant client à la place.

L'ID client est un identifiant unique que vous attribuez à votre appareil ou à votre client. L'ID client est défini dans le protocole MQTT et peut contenir des caractères alphanumériques, des traits de soulignement ou des tirets. Il doit être unique à chaque appareil auquel il se connecte AWS IoT.

> **Note**
>
> - Si votre appareil a été enregistré en tant qu'objet dans le AWS IoT registre, l'ID client peut être le même que le nom de l'objet.

- Si l'exécution de votre commande cible un ID client MQTT spécifique, pour recevoir la charge utile de commande de la rubrique Commandes basées sur l'ID client, votre appareil doit se connecter à AWS IoT l'aide du même identifiant client.

L'ID client est généralement l'ID client MQTT que vos appareils peuvent utiliser pour se connecter AWS IoT Core. Cet identifiant est utilisé AWS IoT pour identifier chaque appareil spécifique et gérer les connexions et les abonnements.

Considérations relatives au délai d'exécution des commandes

Le délai d'attente indique la durée en secondes pendant laquelle votre appareil peut fournir le résultat de l'exécution de la commande.

Une fois que vous avez créé une exécution de commande, un temporisateur démarre. Si l'appareil s'est déconnecté ou n'a pas communiqué le résultat de l'exécution dans le délai imparti, l'exécution de la commande expirera et l'état d'exécution sera signalé sous `TIMED_OUT` la forme.

Ce champ est facultatif et sera défini par défaut sur 10 secondes si vous ne spécifiez aucune valeur. Vous pouvez également configurer le délai d'attente à une valeur maximale de 12 heures.

Valeur du délai d'expiration et statut **`TIMED_OUT`** d'exécution

Un délai d'attente peut être signalé à la fois par le cloud et par l'appareil.

Une fois la commande envoyée à l'appareil, une minuterie démarre. Si aucune réponse n'a été reçue de l'appareil dans le délai spécifié, comme décrit ci-dessus. Dans ce cas, le cloud définit l'état d'exécution de la commande `TIMED_OUT` avec le code de raison comme`$NO_RESPONSE_FROM_DEVICE`.

Cela peut se produire dans l'un des cas suivants.

- L'appareil s'est déconnecté lors de l'exécution de la commande.
- Le périphérique n'a pas pu terminer l'exécution de la commande dans le délai spécifié.
- L'appareil n'a pas communiqué les informations d'état mises à jour dans le délai imparti.

Dans ce cas, lorsque l'état d'exécution de `TIMED_OUT` est signalé depuis le cloud, l'exécution de la commande n'est pas terminale. Votre appareil peut publier une réponse qui remplace le statut par

l'un des états du terminal, `SUCCEEDEDFAILED`, ou. `REJECTED` L'exécution de la commande devient désormais un terminal et n'accepte aucune autre mise à jour.

Votre appareil peut également mettre à jour un `TIMED_OUT` état initié par le cloud en signalant qu'un délai d'attente s'est produit lors de l'exécution de la commande. Dans ce cas, l'état d'exécution de la commande reste `TIMED_OUT` inchangé, mais l'`statusReason`objet sera mis à jour en fonction des informations communiquées par le périphérique. L'exécution de la commande deviendra désormais un terminal et aucune autre mise à jour ne sera acceptée.

Utilisation de sessions persistantes MQTT

Vous pouvez configurer des sessions persistantes MQTT à utiliser avec la fonction de AWS IoT Device Management commandes. Cette fonctionnalité est particulièrement utile lorsque votre appareil est hors ligne et que vous voulez vous assurer que l'appareil reçoit toujours la commande lorsqu'il revient en ligne avant la fin du délai d'expiration, et qu'il exécute les instructions spécifiées.

Par défaut, l'expiration de la session persistante MQTT est fixée à 60 minutes. Si le délai d'exécution de vos commandes est configuré sur une valeur supérieure à cette durée, les exécutions de commandes de plus de 60 minutes peuvent être rejetées par le courtier de messages et peuvent échouer. Pour exécuter des commandes d'une durée supérieure à 60 minutes, vous pouvez demander une augmentation de la durée d'expiration de la session persistante.

> **Note**
>
> Pour vous assurer que vous utilisez correctement la fonctionnalité de sessions persistantes MQTT, assurez-vous que l'indicateur Clean Start est défini sur zéro. Pour plus d'informations, consultez la section Sessions persistantes MQTT.

Lancer l'exécution d'une commande (console)

Pour commencer à exécuter la commande depuis la console, rendez-vous sur la page Command Hub de la AWS IoT console et effectuez les étapes suivantes.

1. Pour exécuter la commande que vous avez créée, choisissez Exécuter la commande.
2. Passez en revue les informations concernant la commande que vous avez créée, le fichier de charge utile et le type de format, ainsi que les rubriques MQTT réservées.
3. Spécifiez le périphérique cible pour lequel vous souhaitez exécuter la commande. L'appareil peut être spécifié comme n'importe AWS IoT quel objet s'il a été enregistré AWS IoT, ou en utilisant

l'ID client si votre appareil n'a pas encore été enregistré. Pour de plus amples informations, consultez Considérations relatives à l'appareil cible.

4. (Facultatif) Configurez une valeur de délai d'expiration pour la commande qui détermine la durée pendant laquelle vous souhaitez que la commande s'exécute avant son expiration. Si votre commande doit s'exécuter pendant plus de 60 minutes, vous devrez peut-être augmenter le délai d'expiration des sessions persistantes MQTT. Pour de plus amples informations, veuillez consulter Considérations relatives au délai d'exécution des commandes.
5. Sélectionnez Run Command (Exécuter la commande).

Lancer l'exécution d'une commande (AWS CLI)

Utilisez l'opération d'API du plan de données `StartCommandExecution`HTTP pour démarrer l'exécution d'une commande. La demande et la réponse de l'API sont corrélées par l'ID d'exécution de la commande. Une fois que l'appareil a terminé d'exécuter la commande, il peut signaler l'état et le résultat de l'exécution au cloud en publiant un message dans la rubrique de réponse aux commandes. Pour un code de réponse personnalisé, les codes d'application que vous possédez peuvent traiter le message de réponse et publier le résultat sur AWS IoT.

Si vos appareils se sont abonnés à la rubrique de demande de commandes, l'`StartCommandExecution`API publiera le message de charge utile dans cette rubrique. La charge utile peut utiliser le format de votre choix. Pour de plus amples informations, veuillez consulter Charge utile de commande.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request/<PayloadFormat>
```

Si le format de charge utile n'est pas JSON ou CBOR, voici le format de la rubrique de demande de commandes.

```
$aws/commands/<devices>/<DeviceID>/executions/+/request
```

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`StartCommandExecution`action.

Dans cet exemple, remplacez :

- *region*avec votre Région AWS, par exemple`ap-south-1`.
- *account-id*avec votre Compte AWS numéro, par exemple*123456789012*.
- *command-id*avec un identifiant unique pour votre AWS IoT commande, tel que*LockDoor*. Si vous souhaitez envoyer plusieurs commandes, vous pouvez les spécifier dans la politique IAM.
- *devices*avec `thing` ou `client` selon que vos appareils ont été enregistrés en tant qu' AWS IoT objets ou sont spécifiés en tant que clients MQTT.
- *device-id*avec votre AWS IoT `thing-name` or`client-id`.

```
{
  "Effect": "Allow",
  "Action": [
      "iot:StartCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Obtenir le point de terminaison du plan de données spécifique au compte

Avant d'exécuter la commande API, vous devez obtenir l'URL du point de terminaison spécifique au compte pour le `iot:Jobs` point de terminaison. Par exemple, si vous exécutez la commande suivante :

```
aws iot describe-endpoint --endpoint-type iot:Jobs
```

Il renverra l'URL du point de terminaison spécifique au compte, comme indiqué dans l'exemple de réponse ci-dessous.

```
{
    "endpointAddress": "<account-specific-prefix>.jobs.iot.<region>.amazonaws.com"
}
```

Démarrer un exemple d'exécution de commande (AWS CLI)

L'exemple suivant montre comment démarrer l'exécution d'une commande à l'aide de cette `start-command-execution` AWS CLI commande.

Dans cet exemple, remplacez :

- *<command-arn>*avec l'ARN de la commande que vous souhaitez exécuter. Vous pouvez obtenir ces informations à partir de la réponse de la commande `create-command` CLI. Par exemple, si vous exécutez la commande pour changer le mode volant, utilisez`arn:aws:iot:`*region*`:`*account-id*`:command/`*SetComfortSteeringMode*.
- *<target-arn>*avec le Thing ARN de l'appareil cible, qui peut être un objet IoT ou un client MQTT, pour lequel vous souhaitez exécuter la commande. Par exemple, si vous exécutez la commande pour le périphérique ciblemy`RegisteredThing`, utilisez`arn:aws:iot:`*region*`:`*account-id*`:thing/`*myRegisteredThing*.
- *<endpoint-url>*avec le point de terminaison spécifique au compte dans lequel vous l'avez obtenuObtenir le point de terminaison du plan de données spécifique au compte, préfixé par. `https://` Par exemple, `https://`*123456789012abcd*`.jobs.iot.`*ap-south-1*`.amazonaws.com`.
- (Facultatif) Vous pouvez également spécifier un paramètre supplémentaire lors de l'exécution de l'opération `StartCommandExecution` d'API. `executionTimeoutSeconds` Ce champ facultatif indique le délai en secondes pendant lequel le périphérique doit terminer l'exécution de la commande. Par défaut, la valeur est de 10 secondes. Lorsque le statut d'exécution de la commande est `CREATED` défini, un temporisateur démarre. Si le résultat de l'exécution de la commande n'est pas reçu avant l'expiration du délai, le statut passe automatiquement à`TIMED_OUT`.

```
aws iot-jobs-data start-command-execution \
    --command-arn <command-arn>  \
    --target-arn <target-arn> \
    --endpoint <endpoint-url> \
    --execution-timeout-seconds 900
```

L'exécution de cette commande renvoie un ID d'exécution de commande. Vous pouvez utiliser cet ID pour demander le statut, les détails et l'historique de l'exécution des commandes.

> **Note**
>
> Si la commande est obsolète, la demande d'`StartCommandExecution`API échouera avec une exception de validation. Pour corriger cette erreur, restaurez d'abord la commande à l'aide de l'`UpdateCommand`API, puis exécutez la `StartCommandExecution` demande.

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542"
}
```

Mettre à jour le résultat de l'exécution d'une commande

Utilisez l'opération d'API du plan de données `UpdateCommandExecution` MQTT pour mettre à jour le statut ou le résultat d'une exécution de commande.

> **Note**
>
> Avant d'utiliser cette API :
>
> - Votre appareil doit avoir établi une connexion MQTT et être abonné aux rubriques de demande et de réponse des commandes. Pour de plus amples informations, veuillez consulter Flux de travail de commandes de haut niveau.
> - Vous devez déjà avoir exécuté cette commande à l'aide de l'opération `StartCommandExecution` API.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM autorise votre appareil à effectuer ces actions. Vous trouverez ci-dessous un exemple de politique qui autorise votre appareil à effectuer cette action. Pour d'autres exemples de politiques IAM autorisant l'utilisateur à effectuer l'`UpdateCommandExecution`action, consultezExemples de stratégies de connexion et de publication.

Dans cet exemple, remplacez :

- *`Region`*avec votre Région AWS, par exemple`ap-south-1`.
- *`AccountID`*avec votre Compte AWS numéro, par exemple*`123456789012`*.
- *`ThingName`*avec le nom de l' AWS IoT objet pour lequel vous ciblez l'exécution de la commande, par exemple*`myRegisteredThing`*.
- *`commands-request-topic`*et *`commands-response-topic`* avec les noms des sujets de demande et de réponse de vos AWS IoT commandes. Pour de plus amples informations, veuillez consulter Flux de travail de commandes de haut niveau.

Exemple de politique IAM pour l'ID client MQTT

Le code suivant montre un exemple de politique de périphérique lors de l'utilisation de l'ID client MQTT.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/request/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/clients/
${iot:ClientId}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
```

```
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/request/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/clients/
${iot:ClientId}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
  ]
}
```

Exemple de politique IAM pour l'IoT

Le code suivant montre un exemple de politique relative aux appareils lors de l'utilisation d'un AWS IoT objet.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": [
```

```
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/request/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topic/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/*/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/request/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/accepted/json",
        "arn:aws:iot:us-east-1:123456789012:topicfilter/$aws/commands/things/
${iot:Connection.Thing.ThingName}/executions/+/response/rejected/json"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/${iot:ClientId}"
    }
  ]
}
```

Comment utiliser l'**`UpdateCommandExecution`**API

Une fois que l'exécution de la commande est reçue sur le sujet de la demande, le dispositif traite la commande. Il utilise ensuite l'`UpdateCommandExecution`API pour mettre à jour l'état et le résultat de l'exécution de la commande en fonction de la rubrique de réponse suivante.

```
$aws/commands/<devices>/<DeviceID>/executions/<ExecutionId>/response/<PayloadFormat>
```

Dans cet exemple, *`<DeviceID>`* il s'agit de l'identifiant unique de votre équipement cible et *`<execution-id>`* de l'identifiant de l'exécution de la commande sur le périphérique cible. Il *`<PayloadFormat>`* peut s'agir de JSON ou de CBOR.

> **Note**
>
> Si vous n'avez pas enregistré votre appareil AWS IoT, vous pouvez utiliser l'identifiant client comme identifiant au lieu d'un nom d'objet.

```
$aws/commands/clients/<ClientID>/executions/<ExecutionId>/response/<PayloadFormat>
```

L'appareil a signalé des mises à jour de l'état d'exécution

Vos appareils peuvent utiliser l'API pour signaler l'une des mises à jour de statut suivantes lors de l'exécution de la commande. Pour plus d'informations sur ces statuts, consultezÉtat de l'exécution de la commande.

- `IN_PROGRESS`: Lorsque l'appareil commence à exécuter la commande, il peut mettre à jour l'état à`IN_PROGRESS`.
- `SUCCEEDED`: Lorsque l'appareil traite avec succès la commande et termine son exécution, il peut publier un message dans le sujet de réponse sous le nom de`SUCCEEDED`.
- `FAILED`: Si l'appareil n'a pas réussi à exécuter la commande, il peut publier un message dans le sujet de réponse sous le nom`FAILED`.
- `REJECTED`: Si l'appareil n'accepte pas la commande, il peut publier un message dans le sujet de réponse en tant que`REJECTED`.
- `TIMED_OUT`: L'état d'exécution de la commande peut changer pour `TIMED_OUT` l'une des raisons suivantes.

- Le résultat de l'exécution de la commande n'a pas été reçu. Cela peut se produire parce que l'exécution n'a pas été terminée dans le délai spécifié ou si l'appareil n'a pas publié les informations d'état dans le sujet de réponse.
- L'appareil signale qu'un délai d'attente s'est produit lors de la tentative d'exécution de la commande.

Pour plus d'informations sur le `TIMED_OUT` statut, consultezValeur du délai d'expiration et statut `TIMED_OUT` d'exécution.

Considérations relatives à l'utilisation de l'**`UpdateCommandExecution`**API

Voici quelques points importants à prendre en compte lors de l'utilisation de l'`UpdateCommandExecution`API.

- Vos appareils peuvent utiliser un `statusReason` objet facultatif, qui peut être utilisé pour fournir des informations supplémentaires sur l'exécution. Si vos appareils fournissent cet objet, le `reasonCode` champ de l'objet est obligatoire, mais le `reasonDescription` champ est facultatif.
- Lorsque vos appareils utilisent l'`statusReason`objet, ils `reasonCode` doivent utiliser le modèle`[A-Z0-9_-]+`, qui ne doit pas dépasser 64 caractères. Si vous fournissez le`reasonDescription`, assurez-vous qu'il ne dépasse pas 1 024 caractères. Il peut utiliser n'importe quel caractère à l'exception des caractères de contrôle tels que les nouvelles lignes.
- Vos appareils peuvent utiliser un `result` objet facultatif pour fournir des informations sur le résultat de l'exécution de la commande, telles que la valeur de retour d'un appel de fonction à distance. Si vous le fournissez`result`, il doit nécessiter au moins une entrée.
- Dans le `result` champ, vous spécifiez les entrées sous forme de paires clé-valeur. Pour chaque entrée, vous devez spécifier les informations de type de données sous forme de chaîne, de booléen ou de binaire. Un type de données chaîne doit utiliser la clés, un type de données booléen utilise la clé b et un type de données binaire doit utiliser la clé. `bin` Vous devez vous assurer que ces types de données sont mentionnés en minuscules.
- Si vous rencontrez une erreur lors de l'exécution de l'`UpdateCommandExecution`API, vous pouvez consulter l'erreur dans le groupe de `AWSIoTLogsV2` journaux d'Amazon CloudWatch. Pour plus d'informations sur l'activation de la journalisation et l'affichage des journaux, consultezConfiguration de la AWS IoT journalisation.

`UpdateCommandExecution`Exemple d'API

Le code suivant montre comment votre appareil peut utiliser l'`UpdateCommandExecution`API pour signaler l'état d'exécution, le `statusReason` champ pour fournir des informations supplémentaires sur l'état et le champ de résultat pour fournir des informations sur le résultat de l'exécution, comme le pourcentage de batterie de la voiture dans ce cas.

```
{
  "status": "IN_PROGRESS",
  "statusReason": {
    "reasonCode": "200",
    "reasonDescription": "Execution_in_progress"
  },
  "result": {
        "car_battery": {
            "s": "car battery at 50 percent"
        }
    }
}
```

Récupérer l'exécution d'une commande

Après avoir exécuté une commande, vous pouvez récupérer des informations sur l'exécution de la commande à partir de la AWS IoT console et à l'aide du AWS CLI. Vous pouvez obtenir les informations suivantes.

> **Note**
>
> Pour récupérer le dernier statut d'exécution des commandes, votre appareil doit publier les informations d'état dans la rubrique de réponse à l'aide de l'API `UpdateCommandExecution` MQTT, comme décrit ci-dessous. Jusqu'à ce que l'appareil publie sur cette rubrique, l'`GetCommandExecution`API signalera l'état sous la forme `CREATED` ou`TIMED_OUT`.

Chaque exécution de commande que vous créez comportera :

- Un ID d'exécution, qui est un identifiant unique de l'exécution de la commande.
- État de l'exécution de la commande. Lorsque vous exécutez la commande sur le périphérique cible, l'exécution de la commande entre dans un `CREATED` état. Il peut ensuite passer à d'autres états d'exécution de commande, comme décrit ci-dessous.

- Le résultat de l'exécution de la commande.
- L'ID de commande unique et le périphérique cible pour lequel les exécutions ont été créées.
- Date de début, qui indique l'heure à laquelle l'exécution de la commande a été créée.

Récupérer une exécution de commande (console)

Vous pouvez récupérer une exécution de commande depuis la console à l'aide de l'une des méthodes suivantes.

- Depuis la page Command Hub

  Accédez à la page Command Hub de la AWS IoT console et effectuez ces étapes.

  1. Choisissez la commande pour laquelle vous avez créé une exécution sur le périphérique cible.
  2. Sur la page des détails des commandes, sous l'onglet Historique des commandes, vous pouvez voir les exécutions que vous avez créées. Choisissez l'exécution pour laquelle vous souhaitez récupérer des informations.
  3. Si vos appareils ont utilisé l'`UpdateCommandExecution`API pour fournir les informations relatives aux résultats, vous pouvez les trouver dans l'onglet Résultats de cette page.
- Depuis la page du hub Thing

  Si vous avez choisi un AWS IoT objet comme équipement cible lors de l'exécution de la commande, vous pouvez consulter les détails de l'exécution sur la page du hub d'objets.

  1. Accédez à la page Thing Hub de la AWS IoT console et choisissez l'objet pour lequel vous avez créé l'exécution de la commande.
  2. Dans la page des détails de l'objet, dans l'historique des commandes, vous pouvez voir les exécutions que vous avez créées. Choisissez l'exécution pour laquelle vous souhaitez récupérer des informations.
  3. Si vos appareils ont utilisé l'`UpdateCommandExecution`API pour fournir les informations relatives aux résultats, vous pouvez les trouver dans l'onglet Résultats de cette page.

Récupérer une exécution de commande (CLI)

Utilisez l'opération d'API HTTP du plan de `GetCommandExecution` AWS IoT Core contrôle pour récupérer les informations relatives à l'exécution d'une commande. Vous devez déjà avoir exécuté cette commande à l'aide de l'opération `StartCommandExecution` API.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`GetCommandExecution`action.

Dans cet exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`ap-south-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple*`123456789012`*.
- *`command-id`*avec votre identifiant de AWS IoT commande unique, tel que*`LockDoor`*.
- *`devices`*avec `thing` ou `client` selon que vos appareils ont été enregistrés en tant qu' AWS IoT objets ou sont spécifiés en tant que clients MQTT.
- *`device-id`*avec votre AWS IoT `thing-name` or`client-id`.

```
{
  "Effect": "Allow",
  "Action": [
      "iot:GetCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
}
```

Récupérer un exemple d'exécution de commande

L'exemple suivant montre comment récupérer des informations sur une commande exécutée à l'aide de cette `start-command-execution` AWS CLI commande. L'exemple suivant montre comment récupérer des informations sur une commande exécutée pour désactiver le mode volant.

Dans cet exemple, remplacez :

- *`<execution-id>`*avec l'identifiant de l'exécution de la commande pour laquelle vous souhaitez récupérer des informations.
- *`<target-arn>`*avec le numéro de ressource Amazon (ARN) de l'appareil pour lequel vous ciblez l'exécution. Vous pouvez obtenir ces informations à partir de la réponse de la commande `start-command-execution` CLI.

- Facultativement, si vos appareils ont utilisé l'`UpdateCommandExection`API pour fournir le résultat de l'exécution, vous pouvez spécifier s'il convient d'inclure le résultat de l'exécution de la commande dans la réponse de l'`GetCommandExecution`API à l'aide de l'`GetCommandExecution`API.

```
aws iot get-command-execution
    --execution-id <execution-id> \
    --target-arn <target-arn> \
    --include-result
```

L'exécution de cette commande génère une réponse contenant des informations sur l'ARN de l'exécution de la commande, son état d'exécution, ainsi que l'heure à laquelle elle a commencé à s'exécuter et à laquelle elle s'est terminée. Il fournit également un `statusReason` objet contenant des informations supplémentaires sur le statut. Pour plus d'informations sur les différents statuts et la raison du statut, consultezÉtat de l'exécution de la commande.

Le code suivant montre un exemple de réponse à la demande d'API.

> **Note**
>
> Le `completedAt` champ de la réponse d'exécution correspond au moment où l'appareil signale l'état du terminal au cloud. En cas d'`TIMED_OUT`état, ce champ ne sera défini que lorsque l'appareil signalera un délai d'expiration. Lorsque le `TIMED_OUT` statut est défini par le cloud, il n'est pas mis à jour. `TIMED_OUT` Pour plus d'informations sur le comportement en cas de temporisation, consultezConsidérations relatives au délai d'exécution des commandes.

```
{
    "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
    "commandArn": "arn:aws:iot:ap-south-1:123456789012:command/LockDoor",
    "targetArn": "arn:aws:iot:ap-south-1:123456789012:thing/myRegisteredThing",
    "status": "SUCCEEDED",
    "statusReason": {
        "reasonCode": "DEVICE_SUCCESSFULLY_EXECUTED",
        "reasonDescription": "SUCCESS"
    },
    "result": {
        "sn": { "s": "ABC-001" },
```

```
        "digital": { "b": true }
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "completedAt": "2024-03-23T00:50:10.095000-07:00"
}
```

Affichage des mises à jour des commandes à l'aide du client de test MQTT

Vous pouvez utiliser le client de test MQTT pour visualiser l'échange de messages via MQTT lorsque vous utilisez la fonction de commandes. Une fois que votre appareil a établi une connexion MQTT avec AWS IoT, vous pouvez créer une commande, spécifier la charge utile, puis l'exécuter sur le périphérique. Lorsque vous exécutez la commande, si votre appareil s'est abonné à la rubrique de demande réservée MQTT pour les commandes, le message de charge utile sera publié dans cette rubrique.

Le dispositif reçoit ensuite les instructions de charge utile et exécute les opérations spécifiées sur le dispositif IoT. Il utilise ensuite l'`UpdateCommandExecution`API pour publier le résultat de l'exécution des commandes et les informations d'état dans les rubriques de réponse réservées aux commandes du MQTT. AWS IoT Device Management écoute les mises à jour sur les sujets de réponse, stocke les informations mises à jour et publie des journaux sur Amazon et AWS CloudTrail Amazon. CloudWatch Vous pouvez ensuite récupérer les dernières informations d'exécution des commandes depuis la console ou à l'aide de l'`GetCommandExecution`API.

Les étapes suivantes montrent comment utiliser le client de test MQTT pour observer les messages.

1. Ouvrez le client de test MQTT dans la AWS IoT console.
2. Dans l'onglet S'abonner, entrez le sujet suivant, puis choisissez S'abonner, où se *<thingId>* trouve le nom de l'appareil avec lequel vous vous êtes enregistré AWS IoT.

> **Note**
>
> Vous pouvez trouver le nom de votre appareil sur la page Thing Hub de la AWS IoT console, ou si vous n'avez pas enregistré votre appareil en tant qu'objet, vous pouvez enregistrer l'appareil lorsque vous vous connectez à AWS IoT partir de la page Connect device.

```
$aws/commands/things/<thingId>/executions/+/request
```

3. (Facultatif) Dans l'onglet S'abonner, vous pouvez également saisir les sujets suivants et choisir S'abonner.

```
$aws/commands/things/+/executions/+/response/accepted/json
$aws/commands/things/+/executions/+/response/rejected/json
```

4. Lorsque vous lancez l'exécution d'une commande, la charge utile du message est envoyée à l'appareil en utilisant le sujet de demande auquel l'appareil s'est abonné,`$aws/commands/things/<thingId>/executions/+/request`. Dans le client de test MQTT, vous devriez voir la charge utile de commande contenant les instructions permettant au périphérique de traiter la commande.

5. Une fois que le périphérique a commencé à exécuter la commande, il peut publier des mises à jour de statut dans la rubrique de réponse réservée MQTT suivante pour les commandes.

```
$aws/commands/<devices>/<device-id>/executions/<executionId>/response/json
```

Par exemple, considérez une commande que vous avez exécutée pour allumer le climatiseur de votre voiture afin de réduire la température à la valeur souhaitée. Le JSON suivant montre un exemple de message publié par le véhicule dans le sujet de réponse qui indique qu'il n'a pas réussi à exécuter la commande.

```
{
  "deviceId": "My_Car",
  "executionId": "07e4b780-7eca-4ffd-b772-b76358da5542",
  "status": "FAILED",
  "statusReason": {
    "reasonCode": "CAR_LOW_ON_BATTERY",
    "reasonDescription": "Car battery is lower than 5 percent"
  }
}
```

Dans ce cas, vous pouvez charger la batterie de votre voiture, puis exécuter à nouveau la commande.

Répertoriez les exécutions de commandes dans votre Compte AWS

Après avoir exécuté une commande, vous pouvez récupérer des informations sur l'exécution de la commande à partir de la AWS IoT console et à l'aide du AWS CLI. Vous pouvez obtenir les informations suivantes.

- Un ID d'exécution, qui est un identifiant unique de l'exécution de la commande.
- État de l'exécution de la commande. Lorsque vous exécutez la commande sur le périphérique cible, l'exécution de la commande entre dans un `CREATED` état. Il peut ensuite passer à d'autres états d'exécution de commande, comme décrit ci-dessous.
- L'ID de commande unique et le périphérique cible pour lequel les exécutions ont été créées.
- Date de début, qui indique l'heure à laquelle l'exécution de la commande a été créée.

Répertorier les exécutions de commandes dans votre compte (console)

Vous pouvez voir toutes les exécutions de commandes depuis la console à l'aide de l'une des méthodes suivantes.

- Depuis la page Command Hub

  Accédez à la page Command Hub de la AWS IoT console et effectuez ces étapes.

  1. Choisissez la commande pour laquelle vous avez créé une exécution sur le périphérique cible.
  2. Sur la page des détails de la commande, accédez à l'onglet Historique des commandes et vous verrez la liste des exécutions que vous avez créées.
- Depuis la page du hub Thing

  Si vous avez choisi un AWS IoT objet comme équipement cible lors de l'exécution de la commande et que vous avez créé plusieurs exécutions de commandes pour un seul appareil, vous pouvez consulter les exécutions de cet appareil sur la page du hub d'objets.

  1. Accédez à la page Thing Hub de la AWS IoT console et choisissez l'objet pour lequel vous avez créé les exécutions.
  2. Sur la page des détails de l'objet, dans l'historique des commandes, vous pouvez voir la liste des exécutions que vous avez créées pour l'appareil.

Répertorier les exécutions de commandes dans votre compte (CLI)

Utilisez l'opération d'API HTTP du plan de `ListCommandExecutions` AWS IoT Core contrôle pour répertorier toutes les exécutions de commandes dans votre compte.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM vous autorise à effectuer cette action sur l'appareil. L'exemple suivant montre une politique IAM qui autorise l'utilisateur à effectuer l'`ListCommandExecutions`action.

Dans cet exemple, remplacez :

- *`region`*avec votre Région AWS, par exemple`ap-south-1`.
- *`account-id`*avec votre Compte AWS numéro, par exemple*`123456789012`*.
- *`command-id`*avec votre identifiant de AWS IoT commande unique, tel que*`LockDoor`*.

```
{
  "Effect": "Allow",
  "Action": "iot:ListCommandExecutions",
  "Resource": *
}
```

Exemple d'exécution de commandes de liste

L'exemple suivant vous montre comment répertorier les exécutions de commandes dans votre Compte AWS.

Lorsque vous exécutez la commande, vous devez spécifier s'il faut filtrer la liste pour afficher uniquement les exécutions de commandes créées pour un appareil particulier à l'aide de`targetArn`, ou les exécutions pour une commande spécifique spécifiée à l'aide du`commandArn`.

Dans cet exemple, remplacez :

- *`<target-arn>`*avec le numéro de ressource Amazon (ARN) de l'appareil pour lequel vous ciblez l'exécution, tel que`arn:aws:iot:`*`us-east-1`*`:`*`123456789012`*`:thing/`*`b8e4157c98f332cffb37627f`*.
- *`<target-arn>`*avec le numéro de ressource Amazon (ARN) de l'appareil pour lequel vous ciblez l'exécution, tel que`arn:aws:iot:`*`us-east-1`*`:`*`123456789012`*`:thing/`*`b8e4157c98f332cffb37627f`*.

- *<after>*avec le délai après lequel vous souhaitez répertorier les exécutions créées, par exemple,`2024-11-01T03:00`.

```
aws iot list-command-executions \
--target-arn <target-arn> \
--started-time-filter '{after=<after>}' \
--sort-order "ASCENDING"
```

L'exécution de cette commande génère une réponse qui contient une liste des exécutions de commandes que vous avez créées, ainsi que l'heure à laquelle les exécutions ont commencé à s'exécuter et à laquelle elles se sont terminées. Il fournit également des informations sur le statut, ainsi que l'`statusReason`objet qui contient des informations supplémentaires sur le statut.

```
{
    "commandExecutions": [
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "b2b654ca-1a71-427f-9669-e74ae9d92d24",
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "TIMED_OUT",
            "createdAt": "2024-11-24T14:39:25.791000-08:00",
            "startedAt": "2024-11-24T14:39:25.791000-08:00"
        },
        {
            "commandArn": "arn:aws:iot:us-east-1:123456789012:command/TestMe002",
            "executionId": "34bf015f-ef0f-4453-acd0-9cca2d42a48f",
            "targetArn": "arn:aws:iot:us-east-1:123456789012:thing/
b8e4157c98f332cffb37627f",
            "status": "IN_PROGRESS",
            "createdAt": "2024-11-24T14:05:36.021000-08:00",
            "startedAt": "2024-11-24T14:05:36.021000-08:00"
        }
    ]
}
```

Pour plus d'informations sur les différents statuts et la raison du statut, consultezÉtat de l'exécution de la commande.

Supprimer l'exécution d'une commande

Si vous ne souhaitez plus utiliser l'exécution d'une commande, vous pouvez la supprimer définitivement de votre compte.

> **Note**
>
> - L'exécution d'une commande ne peut être supprimée que si elle est entrée dans un statut de terminal`SUCCEEDED`, tel que`FAILED`, ou`REJECTED`.
> - Cette opération ne peut être effectuée qu'à l'aide de l' AWS IoT Core API ou du AWS CLI. Il n'est pas disponible depuis la console.

Exemple de politique IAM

Avant d'utiliser cette opération d'API, assurez-vous que votre politique IAM autorise votre appareil à effectuer ces actions. Vous trouverez ci-dessous un exemple de politique qui autorise votre appareil à effectuer cette action.

Dans cet exemple, remplacez :

- *`Region`*avec votre Région AWS, par exemple`ap-south-1`.
- *`AccountID`*avec votre Compte AWS numéro, par exemple*`123456789012`*.
- *`CommandID`*avec l'identifiant de la commande dont vous souhaitez supprimer l'exécution.
- *`devices`*avec `thing` ou `client` selon que vos appareils ont été enregistrés en tant qu' AWS IoT objets ou sont spécifiés en tant que clients MQTT.
- *`device-id`*avec votre AWS IoT `thing-name` or`client-id`.

```
{
  "Effect": "Allow",
  "Action": [
      "iot:DeleteCommandExecution"
  ],
  "Resource": [
      "arn:aws:iot:region:account-id:command/command-id",
      "arn:aws:iot:region:account-id:devices/device-id"
  ]
```

```
}
```

Supprimer un exemple d'exécution de commande

L'exemple suivant montre comment supprimer une commande à l'aide de cette `delete-command` AWS CLI commande. En fonction de votre application, remplacez-le *<execution-id>* par l'identifiant de l'exécution de la commande que vous supprimez, puis *<target-arn>* par l'ARN de votre appareil cible.

```
aws iot delete-command-execution \
--execution-id <execution-id> \
--target-arn <target-arn>
```

Si la demande d'API aboutit, l'exécution de la commande génère un code d'état de 200. Vous pouvez utiliser l'`GetCommandExecution`API pour vérifier que l'exécution de la commande n'existe plus dans votre compte.

Déprécier une ressource de commande

Vous pouvez déprécier une commande pour indiquer qu'elle n'est pas à jour et qu'elle ne doit pas être utilisée. Par exemple, vous pouvez rendre obsolète une commande qui n'est plus activement maintenue, ou vous pouvez créer une commande plus récente avec le même ID de commande mais utilisant des informations de charge utile différentes.

Considérations clés

Voici quelques points importants à prendre en compte lors de la dépréciation d'une commande :

- Lorsque vous désapprouvez une commande, elle n'est pas supprimée. Vous pouvez toujours récupérer la commande à l'aide de son ID de commande et la restaurer si vous souhaitez la réutiliser.
- Si vous essayez de démarrer l'exécution d'une nouvelle commande sur votre machine cible pour une commande obsolète, cela génère une erreur qui vous empêche d'utiliser out-of-date les commandes.
- Pour exécuter une commande obsolète sur votre machine cible, vous devez d'abord la restaurer. Une fois restaurée, la commande devient disponible et peut être utilisée comme commande normale et vous pouvez exécuter des commandes sur le périphérique cible.

- Si vous désapprouvez une commande alors que son exécution est en cours, les exécutions continueront de s'exécuter sur le périphérique cible jusqu'à ce qu'elles soient terminées. Vous pouvez également récupérer le statut des exécutions de commandes.

Déprécier une ressource de commande (console)

Pour désactiver une commande depuis la console, accédez au hub de commande de la AWS IoT console et effectuez les étapes suivantes.

1. Choisissez la commande que vous souhaitez déprécier, puis sous Actions, choisissez Déprécier.
2. Confirmez que vous souhaitez déprécier la commande, puis choisissez Déprécier.

Déprécier une ressource de commande (CLI)

Vous pouvez marquer une commande comme obsolète à l'aide de la CLI`update-command`. Vous devez d'abord déprécier une commande avant de pouvoir la supprimer. Une fois qu'une commande est devenue obsolète, si vous souhaitez l'utiliser, par exemple pour envoyer une exécution de commande à l'appareil cible, vous devez la déprécier.

```
aws iot update-command \
    --command-id <command-id> \
    --deprecated
```

Par exemple, si vous avez déconseillé la *`ACSwitch`* commande que vous avez mise à jour dans l'exemple ci-dessus, le code suivant montre un exemple de sortie de l'exécution de la commande.

```
{
    "commandId": "turnOffAc",
    "deprecated": true,
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00"
}
```

Vérifiez l'heure et le statut de la dépréciation

Vous pouvez utiliser l'opération `GetCommand` d'API pour déterminer si une commande est obsolète et quand elle l'a été pour la dernière fois.

```
aws iot get-command --command-id <turnOffAC>
```

L'exécution de cette commande génère une réponse contenant des informations sur la commande. Vous pouvez obtenir des informations indiquant quand il a été créé et quand il a été déconseillé à l'aide des dernières informations mises à jour. Ces informations peuvent vous aider à déterminer la durée de vie d'une commande et à déterminer si vous souhaitez la supprimer ou la réutiliser. Par exemple, dans l'*turnOffAc*exemple ci-dessus, le code suivant montre un exemple de réponse.

```
{
    "commandId": "turnOffAC",
    "commandArn": "arn:aws:iot:ap-south-1:123456789012:command/turnOffAC",
    "namespace": "AWS-IoT",
    "payload": {
        "content": "testPayload.json",
        "contentType": "application/json"
    },
    "createdAt": "2024-03-23T00:50:10.095000-07:00",
    "lastUpdatedAt": "2024-05-09T23:16:51.370000-07:00",
    "deprecated": false
}
```

Restaurer une ressource de commande

Pour utiliser la `ACSwitch` commande ou pour l'envoyer à votre appareil, vous devez la restaurer.

Pour restaurer une commande depuis la console, accédez au hub de commande de la AWS IoT console, choisissez la commande que vous souhaitez restaurer, puis sous Actions, choisissez Restaurer.

Pour restaurer une commande à l'aide de l' AWS IoT Core API ou du AWS CLI, utilisez l'opération `UpdateCommand` API ou la `update-command` CLI. Le code suivant montre un exemple de demande et de réponse.

```
aws iot update-command \
    --command-id <command-id>
    --no-deprecated
```

Le code suivant montre un exemple de sortie.

```
{
    "commandId": "ACSwitch",
    "deprecated": false,
```

```
    "lastUpdatedAt": "2024-05-09T23:17:21.954000-07:00"
}
```

AWS IoT tunneling sécurisé

Lorsque des appareils sont déployés derrière des pare-feu restreints sur des sites distants, vous devez pouvoir accéder à ces appareils pour le dépannage, les mises à jour de configuration et d'autres tâches opérationnelles. Utilisez le tunneling sécurisé pour établir une communication bidirectionnelle avec des appareils distants via une connexion sécurisée gérée par. AWS IoT Le tunneling sécurisé ne nécessite pas de mise à jour de vos règles de pare-feu entrantes, ce qui vous permet de conserver le même niveau de sécurité que celui fourni par les règles de pare-feu sur un site distant.

Prenons l'exemple d'un capteur situé dans une usine à quelques centaines de kilomètres de distance et qui ne parvient pas à mesurer la température de l'usine. Vous pouvez utiliser le tunneling sécurisé pour ouvrir et démarrer rapidement une session sur ce capteur. Après avoir identifié le problème (par exemple, un fichier de configuration incorrect), vous pouvez réinitialiser le fichier et redémarrer le capteur via la même session. Par rapport à un dépannage plus traditionnel (par exemple, envoyer un technicien à l'usine pour vérifier le capteur), le tunneling sécurisé réduit le temps de réponse aux incidents et le temps de récupération ainsi que les coûts d'exploitation.

Qu'est-ce que le tunneling sécurisé ?

Utilisez le tunneling sécurisé pour accéder aux appareils déployés derrière des pare-feux à port restreint sur des sites distants. Vous pouvez vous connecter à l'appareil de destination depuis votre ordinateur portable ou de bureau en tant que périphérique source à l'aide du AWS Cloud. La source et la destination communiquent à l'aide d'un proxy local open source qui s'exécute sur chaque appareil. Le proxy local communique avec le AWS Cloud en utilisant un port ouvert autorisé par le pare-feu, généralement le port 443. Les données transmises par le tunnel sont chiffrées à l'aide du protocole TLS (Transported Layer Security).

Rubriques

- Concepts de tunneling sécurisés
- Comment fonctionne le tunneling sécurisé
- Cycle de vie des tunnels sécurisés

Concepts de tunneling sécurisés

Les termes suivants sont utilisés par le tunneling sécurisé lors de l'établissement d'une communication avec des appareils distants. Pour savoir comment configurer un tunnel sécurisé, consultez Comment fonctionne le tunneling sécurisé.

Jeton d'accès client (CAT)

Une paire de jetons générée par le tunneling sécurisé lors de la création d'un nouveau tunnel. Le CAT est utilisé par les appareils source et de destination pour se connecter au service de tunneling sécurisé. Le CAT ne peut être utilisé qu'une seule fois pour se connecter au tunnel. Pour vous reconnecter au tunnel, faites pivoter les jetons d'accès client à l'aide de l'opération RotateTunnelAccessTokenAPI ou de la commande rotate-tunnel-access-tokenCLI.

Jeton client

Une valeur unique générée par le client que le tunneling AWS IoT sécurisé peut utiliser pour toutes les nouvelles connexions ultérieures au même tunnel. Ce champ est facultatif. Si le jeton client n'est pas fourni, le jeton d'accès client (CAT) ne peut être utilisé qu'une seule fois pour le même tunnel. Les tentatives de connexion ultérieures utilisant le même CAT seront rejetées. Pour plus d'informations sur l'utilisation des jetons client, consultez l'implémentation de référence du proxy local dans GitHub.

Application de destination

Application qui s'exécute sur l'appareil de destination. L'application de destination peut être, par exemple, un démon SSH permettant d'établir une session SSH à l'aide du tunneling sécurisé.

Application de destination

Appareil distant auquel vous souhaitez accéder.

Agent d'appareil

Une application IoT qui se connecte à la passerelle de l' AWS IoT appareil et écoute les nouvelles notifications du tunnel via MQTT. Pour de plus amples informations, veuillez consulter Extrait de l'agent IoT.

Proxy local

Un proxy logiciel qui s'exécute sur les appareils source et destination et relaie un flux de données entre le tunnel sécurisé et l'application de l'appareil. Le proxy local peut être exécuté en mode source ou en mode destination. Pour de plus amples informations, veuillez consulter Proxy local.

Appareil source

Appareil qu'un opérateur utilise pour lancer une session vers l'appareil de destination, généralement un ordinateur portable ou un ordinateur de bureau.

Tunnel

Un chemin logique AWS IoT permettant une communication bidirectionnelle entre un périphérique source et un périphérique de destination.

Comment fonctionne le tunneling sécurisé

Ce qui suit montre comment le tunneling sécurisé établit une connexion entre votre appareil source et votre appareil de destination. Pour plus d'informations sur les différents termes tels que jeton d'accès client (CAT), consultez Concepts de tunneling sécurisés.

1. Ouvrir un tunnel

   Pour ouvrir un tunnel afin de lancer une session avec votre appareil de destination distant, vous pouvez utiliser la AWS Management Console commande AWS CLI open-tunnel ou l'OpenTunnelAPI.

2. Téléchargez la paire de jetons d'accès client

   Après avoir ouvert un tunnel, vous pouvez télécharger le jeton d'accès client (CAT) pour votre source et votre destination et l'enregistrer sur votre appareil source. Vous devez récupérer le CAT et l'enregistrer maintenant avant de démarrer le proxy local.

3. Démarrer le proxy local en mode destination

   L'agent IoT qui a été installé et qui s'exécute sur votre appareil de destination sera abonné au sujet MQTT réservé `$aws/things/thing-name/tunnels/notify` et recevra le CAT. *`thing-name`*Voici le nom de l' AWS IoT objet que vous créez pour votre destination. Pour de plus amples informations, veuillez consulter Rubriques liées au tunneling sécurisé.

   L'agent IoT utilise ensuite le CAT pour démarrer le proxy local en mode destination et établir une connexion du côté destination du tunnel. Pour de plus amples informations, veuillez consulter Extrait de l'agent IoT.

4. Démarrer le proxy local en mode source

Une fois le tunnel ouvert, AWS IoT Device Management fournit le CAT de la source que vous pouvez télécharger sur le périphérique source. Vous pouvez utiliser le CAT pour démarrer le proxy local en mode source, qui connecte ensuite le côté source du tunnel. Pour plus d'informations concernant les proxys locaux, consultez Proxy local.

5. Ouvrez une session SSH

   Les deux côtés du tunnel étant connectés, vous pouvez démarrer une session SSH en utilisant le proxy local du côté source.

Pour plus d'informations sur l'utilisation du AWS Management Console pour ouvrir un tunnel et démarrer une session SSH, consultezOuvrez un tunnel et démarrez une session SSH sur un appareil distant.

La vidéo suivante décrit le fonctionnement du tunneling sécurisé et vous explique le processus de configuration d'une session SSH sur un appareil Raspberry Pi.

Cycle de vie des tunnels sécurisés

Les tunnels peuvent avoir le statut `OPEN` ou `CLOSED`. Les connexions au tunnel peuvent avoir le statut `CONNECTED` ou `DISCONNECTED`. Ce qui suit montre comment fonctionnent les différents statuts de tunnel et de connexion.

1. Lorsque vous ouvrez un tunnel, son état est `OPEN`. L'état de connexion source et de destination du tunnel est défini sur `DISCONNECTED`.
2. Lorsqu'un appareil (source ou destination) se connecte au tunnel, l'état de la connexion correspondante devient `CONNECTED`.
3. Lorsqu'un appareil se déconnecte du tunnel alors que l'état du tunnel reste `OPEN`, l'état de la connexion correspondante redevient `DISCONNECTED`. Un appareil peut se connecter à un tunnel et s'en déconnecter à plusieurs reprises tant que l'état du tunnel reste `OPEN`.

> **Note**
>
> Les jetons d'accès client (CAT) ne peuvent être utilisés qu'une seule fois pour se connecter à un tunnel. Pour vous reconnecter au tunnel, faites pivoter les jetons d'accès client à l'aide de l'opération RotateTunnelAccessTokenAPI ou de la commande rotate-tunnel-access-tokenCLI. Pour obtenir des exemples, consultez Résolution des problèmes

de connectivité liés au tunneling AWS IoT sécurisé en faisant pivoter les jetons d'accès client.

4. `CloseTunnel` Lorsque vous appelez `OPEN` ou que l'état du tunnel reste `MaxLifetimeTimeout` pendant plus longtemps que la valeur, l'état d'un tunnel devient `CLOSED`. Vous pouvez configurer `MaxLifetimeTimeout` lorsque vous appelez `OpenTunnel`. Si vous ne spécifiez pas de valeur, `MaxLifetimeTimeout` est défini par défaut sur 12 heures.

> **Note**
>
> Un tunnel ne peut pas être rouvert lorsque son état est `CLOSED`.

5. Vous pouvez appeler `DescribeTunnel` et `ListTunnels` pour consulter les métadonnées du tunnel lorsque celui-ci est visible. Le tunnel peut être visible dans la AWS IoT console pendant au moins trois heures avant d'être supprimé.

AWS IoT didacticiels de tunneling sécurisé

AWS IoT le tunneling sécurisé aide les clients à établir une communication bidirectionnelle avec les appareils distants situés derrière un pare-feu via une connexion sécurisée gérée par. AWS IoT

Pour faire une démonstration de tunneling AWS IoT sécurisé, utilisez notre démo de tunneling AWS IoT sécurisé sur. GitHub

Les didacticiels suivants vous aideront à apprendre à démarrer et à utiliser le tunneling sécurisé. Vous allez apprendre comment :

1. Créez un tunnel sécurisé à l'aide des méthodes de configuration rapide et manuelle pour accéder à l'appareil distant.
2. Configurez le proxy local lorsque vous utilisez la méthode de configuration manuelle et connectez-vous au tunnel pour accéder au périphérique de destination.
3. Connectez-vous en SSH à l'appareil distant depuis un navigateur sans avoir à configurer le proxy local.
4. Convertissez un tunnel créé à l'aide de AWS CLI ou à l'aide de la méthode de configuration manuelle pour utiliser la méthode de configuration rapide.

Didacticiels dans cette section

Les didacticiels de cette section se concentrent sur la création d'un tunnel à l'aide de la référence AWS Management Console et de l' AWS IoT API. Dans la AWS IoT console, vous pouvez créer un tunnel depuis la page du hub Tunnels ou depuis la page de détails d'un objet que vous avez créé. Pour de plus amples informations, veuillez consulter Méthodes de création de tunnels dans AWS IoT la console.

Vous trouverez ci-dessous les didacticiels de cette section :

- Ouvrez un tunnel et utilisez le protocole SSH basé sur un navigateur pour accéder à un appareil distant

  Ce didacticiel explique comment ouvrir un tunnel depuis la page du hub Tunnels à l'aide de la méthode de configuration rapide. Vous apprendrez également à utiliser le protocole SSH basé sur un navigateur pour accéder au périphérique distant à l'aide d'une interface de ligne de commande contextuelle intégrée à la console. AWS IoT

- Ouvrez un tunnel à l'aide de la configuration manuelle et connectez-vous à un appareil distant

  Ce didacticiel explique comment ouvrir un tunnel depuis la page du hub Tunnels à l'aide de la méthode de configuration manuelle. Vous apprendrez également à configurer et à démarrer le proxy local à partir d'un terminal de votre appareil source et à vous connecter au tunnel.

- Ouvrez un tunnel pour un appareil distant et utilisez le protocole SSH basé sur un navigateur

  Ce didacticiel explique comment ouvrir un tunnel à partir de la page de détails d'un objet que vous avez créé. Vous allez apprendre à créer un nouveau tunnel et à utiliser un tunnel existant. Le tunnel existant correspond au tunnel ouvert le plus récent créé pour l'appareil. Vous pouvez également utiliser le SSH basé sur le navigateur pour accéder à l'appareil distant.

AWS IoT didacticiels de tunneling sécurisé

- Ouvrez un tunnel et démarrez une session SSH sur un appareil distant
- Ouvrez un tunnel pour un appareil distant et utilisez le protocole SSH basé sur un navigateur

Ouvrez un tunnel et démarrez une session SSH sur un appareil distant

Dans ces didacticiels, vous allez apprendre à accéder à distance à un appareil situé derrière un pare-feu. Vous ne pouvez pas démarrer une session SSH directe sur l'appareil car le pare-feu bloque tout le trafic entrant. Les didacticiels vous montrent comment ouvrir un tunnel, puis l'utiliser pour démarrer une session SSH sur un appareil distant.

Prérequis pour le didacticiel

Les conditions préalables à l'exécution du didacticiel peuvent varier selon que vous utilisez les méthodes de configuration manuelle ou rapide pour ouvrir un tunnel et accéder à l'appareil distant.

> **Note**
>
> Pour les deux méthodes de configuration, vous devez autoriser le trafic sortant sur le port 443.

- Pour plus d'informations sur les conditions requises pour le didacticiel de la méthode de configuration rapide, voir Conditions préalables à la méthode de configuration rapide.
- Pour plus d'informations sur les conditions requises pour le didacticiel de la méthode de configuration manuelle, voir Conditions préalables à la méthode de configuration manuelle. Si vous utilisez cette méthode de configuration, vous devez configurer le proxy local sur votre appareil source. Pour télécharger le code source du proxy local, voir Implémentation de référence du proxy local sur GitHub.

Méthodes de configuration des tunnels

Dans ces didacticiels, vous découvrirez les méthodes de configuration manuelle et rapide pour ouvrir un tunnel et vous connecter à l'appareil distant. Le tableau suivant montre la différence entre les méthodes de configuration. Après avoir créé le tunnel, vous pouvez utiliser une interface de ligne de commande intégrée au navigateur pour accéder au périphérique distant par SSH. Si vous égarez les jetons ou si le tunnel est déconnecté, vous pouvez envoyer de nouveaux jetons d'accès pour vous reconnecter au tunnel.

Méthodes de configuration rapides et manuelles

Critères	Configuration rapide	Configuration manuelle
Création de tunnel	Créez un nouveau tunnel avec des configurations modifiables par défaut. Pour accéder à votre appareil distant, vous ne pouvez utiliser SSH que comme service de destination.	Créez un tunnel en spécifiant manuellement les configurations du tunnel. Vous pouvez utiliser cette méthode pour vous connecter au périphérique distant à l'aide de services autres que SSH.
Jetons d'accès	Le jeton d'accès à la destination sera automatiquement envoyé à votre appareil sur le thème MQTT réservé, si un nom d'objet est spécifié lors de la création du tunnel. Vous n'êtes pas obligé de télécharger ou de gérer le jeton sur votre appareil source.	Vous devrez télécharger et gérer manuellement le jeton sur votre appareil source. Le jeton d'accès à la destination est automatiquement envoyé à l'apparei l distant sur le sujet MQTT réservé, si un nom d'objet est spécifié lors de la création du tunnel.
Proxy local	Un proxy local basé sur le Web est automatiquement configuré pour que vous puissiez interagir avec l'appareil. Vous n'avez pas besoin de configurer manuellement le proxy local.	Vous devrez configurer et lancer manuellem ent le proxy local. Pour configurer le proxy local, vous pouvez soit utiliser le client de AWS IoT périphérique, soit télécharger l'implémentation de référence du proxy local sur GitHub.

Méthodes de création de tunnels dans AWS IoT la console

Les didacticiels de cette section vous montrent comment créer un tunnel à l'aide de l'API AWS Management Console et de l'OpenTunnelAPI. Si vous configurez la destination lors de la création d'un tunnel, le tunneling AWS IoT sécurisé fournit le jeton d'accès au client de destination au périphérique distant via MQTT (et la rubrique MQTT réservée,). `$aws/things/RemoteDeviceA/tunnels/notify` À la réception du message MQTT, l'agent IoT de l'appareil distant lance le proxy local en mode destination. Pour de plus amples informations, veuillez consulter Rubriques réservées.

> **Note**
>
> Vous pouvez omettre la configuration de destination si vous souhaitez transmettre le jeton d'accès client de destination à l'appareil distant via une autre méthode. Pour de plus amples informations, veuillez consulter Configuration d'un appareil distant et utilisation de l'agent IoT.

Dans la AWS IoT console, vous pouvez créer un tunnel en utilisant l'une des méthodes suivantes. Pour plus d'informations sur les didacticiels qui vous aideront à apprendre à créer un tunnel à l'aide de ces méthodes, consultezDidacticiels dans cette section.

- Hub de tunnels

  Lorsque vous créez le tunnel, vous pouvez spécifier si vous souhaitez utiliser la méthode de configuration rapide ou manuelle pour créer le tunnel et fournir les détails de configuration facultatifs du tunnel. Les détails de configuration incluent également le nom de l'appareil de destination et le service que vous souhaitez utiliser pour vous connecter à l'appareil. Après avoir créé un tunnel, vous pouvez utiliser SSH dans le navigateur ou ouvrir un terminal en dehors de la AWS IoT console pour accéder à votre appareil distant.

- Page de détails de l'objet

  Lorsque vous créez le tunnel, vous pouvez également spécifier si vous souhaitez utiliser le tunnel ouvert le plus récent ou créer un nouveau tunnel pour le périphérique, en plus de choisir les méthodes de configuration et de fournir les détails de configuration du tunnel facultatifs. Vous ne pouvez pas modifier les détails de configuration d'un tunnel existant. Vous pouvez utiliser la méthode de configuration rapide pour faire pivoter les jetons d'accès et le SSH vers le périphérique distant dans le navigateur. Pour ouvrir un tunnel à l'aide de cette méthode, vous devez avoir créé un objet IoT (par exemple`RemoteDeviceA`) dans le AWS IoT registre. Pour plus d'informations, voir Enregistrer un appareil dans le AWS IoT registre.

Didacticiels dans cette section

- Ouvrez un tunnel et utilisez le protocole SSH basé sur un navigateur pour accéder à un appareil distant
- Ouvrez un tunnel à l'aide de la configuration manuelle et connectez-vous à un appareil distant

Ouvrez un tunnel et utilisez le protocole SSH basé sur un navigateur pour accéder à un appareil distant

Vous pouvez utiliser la méthode de configuration rapide ou manuelle pour créer un tunnel. Ce didacticiel explique comment ouvrir un tunnel à l'aide de la méthode de configuration rapide et utiliser le SSH basé sur un navigateur pour se connecter au périphérique distant. Pour obtenir un exemple pratique illustrant la façon d'ouvrir un tunnel à l'aide de la méthode de configuration manuelle, consultezOuvrez un tunnel à l'aide de la configuration manuelle et connectez-vous à un appareil distant.

À l'aide de la méthode de configuration rapide, vous pouvez créer un nouveau tunnel avec des configurations par défaut modifiables. Un proxy local basé sur le Web est configuré pour vous et le jeton d'accès est automatiquement envoyé à votre appareil de destination distant à l'aide de MQTT. Après avoir créé un tunnel, vous pouvez commencer à interagir avec votre appareil distant à l'aide d'une interface de ligne de commande intégrée à la console.

Avec la méthode de configuration rapide, vous devez utiliser SSH comme service de destination pour accéder au périphérique distant. Pour plus d'informations sur les différentes méthodes de lancement, consultez Méthodes de configuration des tunnels.

Conditions préalables à la méthode de configuration rapide

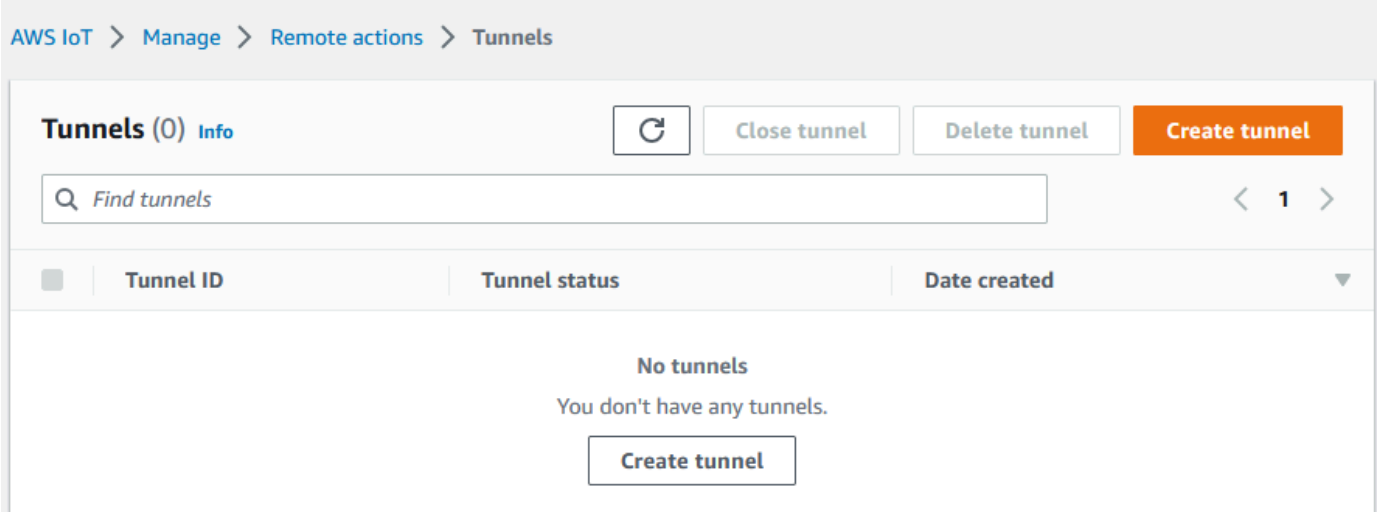
- Les pare-feu derrière l'appareil distant doivent autoriser permettre le trafic sortant sur le port 443. Le tunnel que vous créez utilisera ce port pour se connecter au périphérique distant.
- Vous disposez d'un agent pour appareils IoT (voirExtrait de l'agent IoT) exécuté sur l'appareil distant qui se connecte à la passerelle de l' AWS IoT appareil et qui est configuré avec un abonnement à une rubrique MQTT. Pour plus d'informations, voir connecter un appareil à la passerelle de AWS IoT périphériques.
- Vous devez disposer d'un démon SSH s'exécutant sur l'appareil distant.

Ouvrir un tunnel

Vous pouvez ouvrir un tunnel sécurisé à l'aide du AWS Management Console, de la référence d' AWS IoT API ou du AWS CLI. Vous pouvez éventuellement configurer un nom de destination, mais cela n'est pas obligatoire pour ce didacticiel. Si vous configurez la destination, le tunneling sécurisé fournira automatiquement le jeton d'accès au périphérique distant à l'aide de MQTT. Pour de plus amples informations, veuillez consulter Méthodes de création de tunnels dans AWS IoT la console.

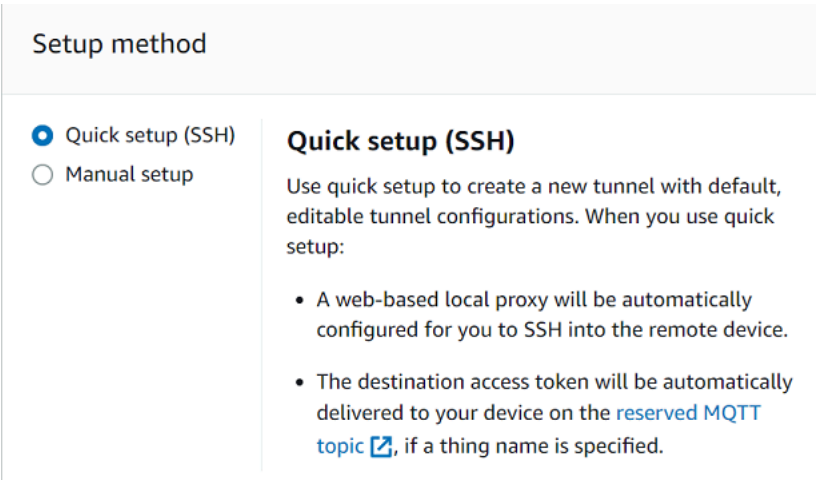
Pour ouvrir un tunnel à l'aide de la console

1. Accédez au hub Tunnels de la AWS IoT console et choisissez Create tunnel.



2. Pour ce didacticiel, choisissez Configuration rapide comme méthode de création de tunnel, puis cliquez sur Suivant.

> **Note**
>
> Si vous créez un tunnel sécurisé à partir de la page de détails d'un objet que vous avez créé, vous pouvez choisir de créer un nouveau tunnel ou d'utiliser un tunnel existant. Pour de plus amples informations, veuillez consulter Ouvrez un tunnel pour un appareil distant et utilisez le protocole SSH basé sur un navigateur.

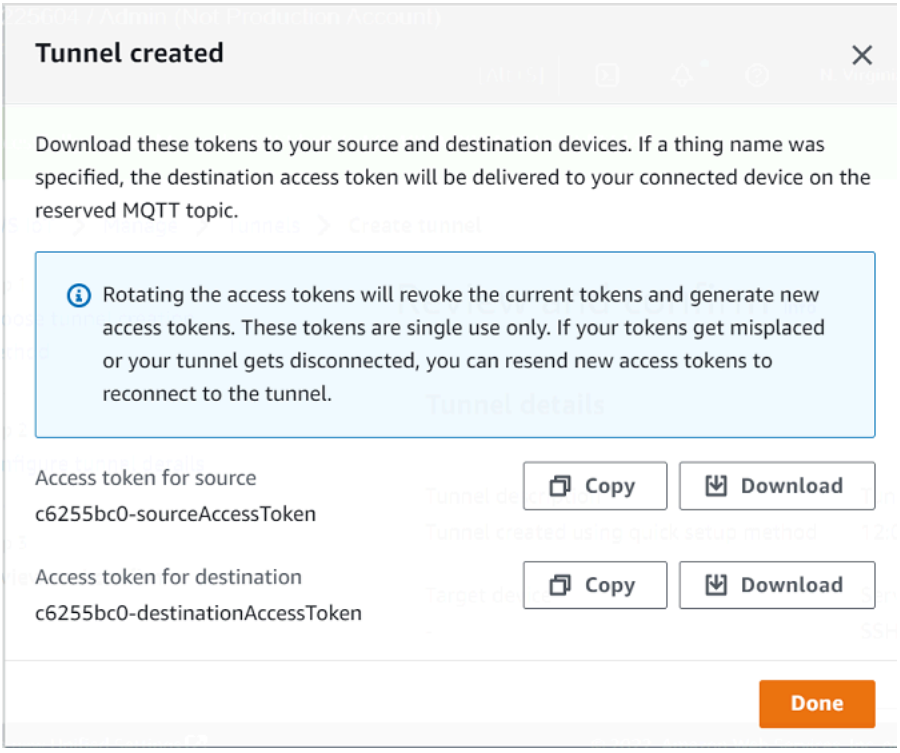


3. Vérifiez et confirmez les détails de configuration du tunnel. Pour créer un tunnel, choisissez Confirmer et créez. Si vous souhaitez modifier ces informations, choisissez Précédent pour revenir à la page précédente, puis confirmez et créez le tunnel.

   > **Note**
   >
   > Lors de l'utilisation de la configuration rapide, le nom du service ne peut pas être modifié. Vous devez utiliser SSH en tant que service.

4. Pour créer le tunnel, choisissez OK.

   Pour ce didacticiel, vous n'avez pas besoin de télécharger les jetons d'accès à la source ou à la destination. Ces jetons ne peuvent être utilisés qu'une seule fois pour se connecter au tunnel. Si votre tunnel est déconnecté, vous pouvez générer et envoyer de nouveaux jetons à votre appareil distant pour vous reconnecter au tunnel. Pour de plus amples informations, veuillez consulter Renvoyer les jetons d'accès au tunnel.



Pour ouvrir un tunnel à l'aide de l'API

Pour ouvrir un nouveau tunnel, vous pouvez utiliser l'opération OpenTunnelAPI.

> **Note**
>
> Vous pouvez créer un tunnel à l'aide de la méthode de configuration rapide uniquement à partir de la AWS IoT console. Lorsque vous utilisez l' AWS IoT API de référence API ou l'API AWS CLI, elle utilisera la méthode de configuration manuelle. Vous pouvez ouvrir le tunnel existant que vous avez créé, puis modifier la méthode de configuration du tunnel pour utiliser la configuration rapide. Pour de plus amples informations, veuillez consulter Ouvrez un tunnel existant et utilisez le protocole SSH basé sur un navigateur.

Voici un exemple de la façon d'exécuter cette opération d'API. Si vous souhaitez spécifier le nom de l'objet et le service de destination, vous pouvez éventuellement utiliser le `DestinationConfig` paramètre. Pour obtenir un exemple pratique illustrant la façon d'utiliser ce paramètre, consultezOuvrez un nouveau tunnel pour le périphérique distant.

```
aws iotsecuretunneling open-tunnel
```

L'exécution de cette commande crée un nouveau tunnel et vous fournit les jetons d'accès à la source et à la destination.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

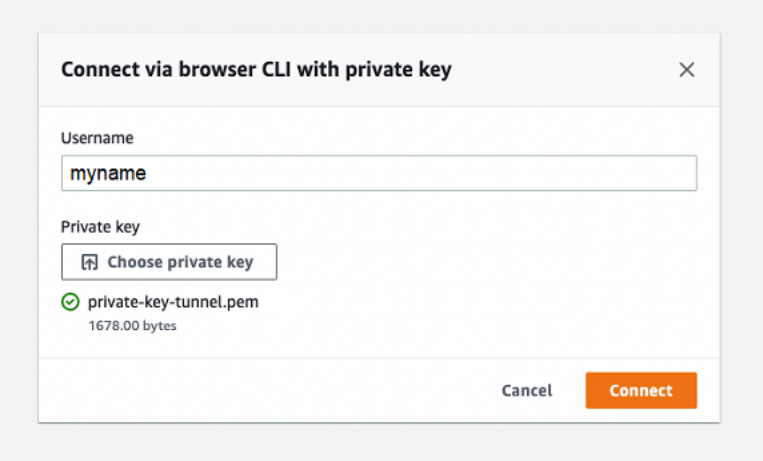
Utilisation du SSH basé sur un navigateur

Une fois que vous avez créé un tunnel à l'aide de la méthode de configuration rapide et que votre appareil de destination s'est connecté au tunnel, vous pouvez accéder au périphérique distant à l'aide d'un SSH basé sur un navigateur. À l'aide du protocole SSH basé sur un navigateur, vous pouvez communiquer directement avec le périphérique distant en saisissant des commandes dans une interface de ligne de commande contextuelle au sein de la console. Cette fonctionnalité facilite l'interaction avec le périphérique distant, car il n'est pas nécessaire d'ouvrir un terminal en dehors de la console ou de configurer le proxy local.

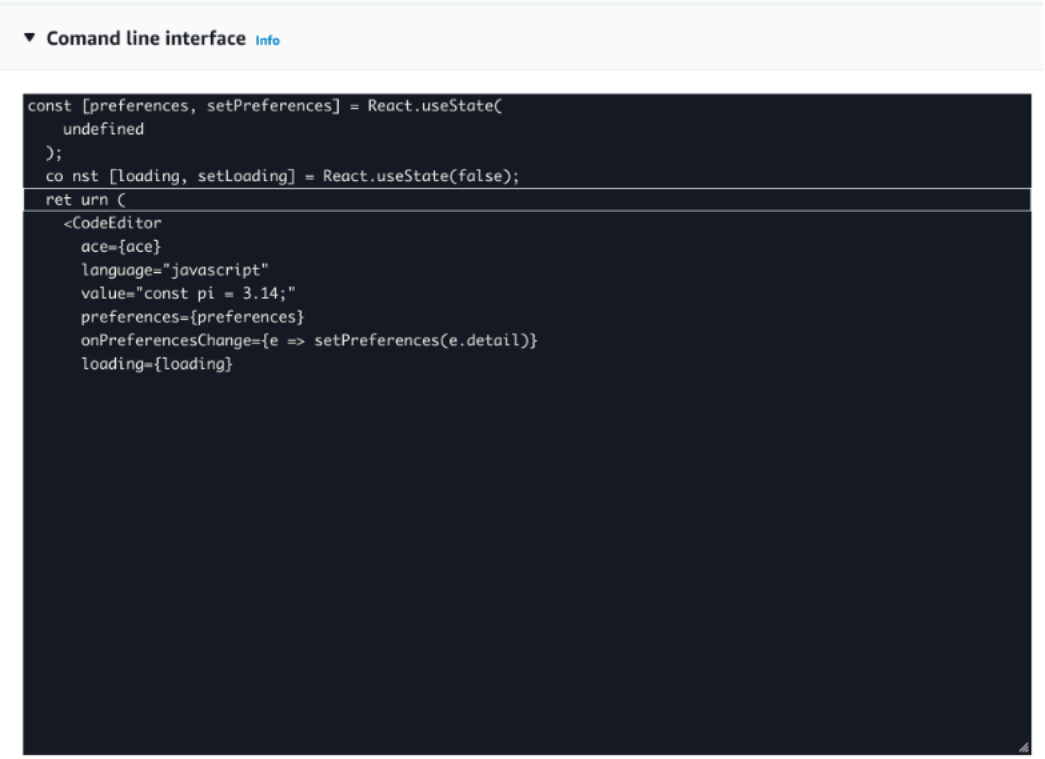
Pour utiliser le SSH basé sur un navigateur

1. Accédez au hub Tunnels de la AWS IoT console et choisissez le tunnel que vous avez créé pour en afficher les détails.
2. Développez la section Secure Shell (SSH), puis choisissez Connect.
3. Choisissez si vous souhaitez vous authentifier dans la connexion SSH en fournissant votre nom d'utilisateur et votre mot de passe ou, pour une authentification plus sécurisée, vous pouvez utiliser la clé privée de votre appareil. Si vous vous authentifiez à l'aide de la clé privée, notez que seules les clés RSA au format PEM (256 et 512) fonctionnent avec la console SSH. AWS IoT Secure Tunneling

   - Pour vous connecter à l'aide de votre nom d'utilisateur et de votre mot de passe, choisissez Utiliser un mot de passe. Vous pouvez ensuite saisir votre nom d'utilisateur et votre mot de passe et commencer à utiliser la CLI intégrée au navigateur.
   - Pour vous connecter à l'aide de la clé privée de votre appareil de destination, choisissez Utiliser la clé privée. Spécifiez votre nom d'utilisateur et téléchargez le fichier de clé privée de l'appareil, puis choisissez Connect pour commencer à utiliser la CLI intégrée au navigateur.



Après vous être authentifié dans la connexion SSH, vous pouvez rapidement commencer à saisir des commandes et à interagir avec le périphérique à l'aide de la CLI du navigateur, car le proxy local a déjà été configuré pour vous.



Si la CLI du navigateur reste ouverte après la durée du tunnel, il se peut qu'elle expire, entraînant la déconnexion de l'interface de ligne de commande. Vous pouvez dupliquer le tunnel et démarrer une autre session pour interagir avec le périphérique distant dans la console elle-même.

Résolution des problèmes liés à l'utilisation du SSH basé sur un navigateur

Ce qui suit montre comment résoudre certains problèmes que vous pourriez rencontrer lors de l'utilisation du SSH basé sur un navigateur.

- Vous voyez une erreur au lieu de l'interface de ligne de commande

  Le message d'erreur s'affiche peut-être parce que votre appareil de destination a été déconnecté. Vous pouvez choisir Générer de nouveaux jetons d'accès pour générer de nouveaux jetons d'accès et envoyer les jetons à votre appareil distant à l'aide de MQTT. Les nouveaux jetons peuvent être utilisés pour se reconnecter au tunnel. La reconnexion au tunnel efface l'historique et actualise la session de ligne de commande.
- Une erreur de déconnexion du tunnel s'affiche lors de l'authentification à l'aide d'une clé privée

  Le message d'erreur s'affiche peut-être parce que votre clé privée n'a peut-être pas été acceptée par l'appareil de destination. Pour résoudre cette erreur, vérifiez le fichier de clé privée que vous avez chargé pour vous authentifier. Si le message d'erreur persiste, consultez les journaux de

votre appareil. Vous pouvez également essayer de vous reconnecter au tunnel en envoyant de nouveaux jetons d'accès à votre appareil distant.

- Votre tunnel a été fermé lors de l'utilisation de la session

  Si votre tunnel a été fermé parce qu'il est resté ouvert pendant une durée supérieure à la durée spécifiée, il est possible que votre session de ligne de commande soit déconnectée. Un tunnel ne peut pas être rouvert une fois fermé. Pour vous reconnecter, vous devez ouvrir un autre tunnel vers l'appareil.

  Vous pouvez dupliquer un tunnel pour créer un nouveau tunnel avec les mêmes configurations que le tunnel fermé. Vous pouvez dupliquer un tunnel fermé depuis la AWS IoT console. Pour dupliquer le tunnel, choisissez le tunnel qui a été fermé pour afficher ses détails, puis choisissez Dupliquer le tunnel. Spécifiez la durée du tunnel que vous souhaitez utiliser, puis créez le nouveau tunnel.

Nettoyage

- Fermer Tunnel

  Nous vous recommandons de fermer le tunnel une fois que vous avez fini de l'utiliser. Un tunnel peut également être fermé s'il est resté ouvert plus longtemps que la durée de tunnel spécifiée. Un tunnel ne peut pas être rouvert une fois fermé. Vous pouvez toujours dupliquer un tunnel en choisissant le tunnel fermé, puis en choisissant Dupliquer le tunnel. Spécifiez la durée du tunnel que vous souhaitez utiliser, puis créez le nouveau tunnel.

  - Pour fermer un tunnel individuel ou plusieurs tunnels depuis la AWS IoT console, accédez au hub Tunnels, choisissez les tunnels que vous souhaitez fermer, puis cliquez sur Fermer le tunnel.
  - Pour fermer un tunnel individuel ou plusieurs tunnels à l'aide de l' AWS IoT API de référence API, utilisez l'CloseTunnelAPI.

    ```
    aws iotsecuretunneling close-tunnel \
        --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    ```

- Supprimer le tunnel

  Vous pouvez supprimer définitivement un tunnel de votre Compte AWS.

> **Warning**
>
> Les actions de suppression sont permanentes et ne peuvent être annulées.

- Pour supprimer un ou plusieurs tunnels de la AWS IoT console, accédez au hub Tunnels, choisissez les tunnels que vous souhaitez supprimer, puis sélectionnez Supprimer le tunnel.
- Pour supprimer un ou plusieurs tunnels à l'aide de l' AWS IoT API de référence de l'API, utilisez l'CloseTunnelAPI. Lorsque vous utilisez l'API, définissez l'`delete`indicateur sur`true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Ouvrez un tunnel à l'aide de la configuration manuelle et connectez-vous à un appareil distant

Lorsque vous ouvrez un tunnel, vous pouvez choisir la méthode de configuration rapide ou manuelle pour ouvrir un tunnel vers le périphérique distant. Ce didacticiel explique comment ouvrir un tunnel à l'aide de la méthode de configuration manuelle et comment configurer et démarrer le proxy local pour se connecter au périphérique distant.

Lorsque vous utilisez la méthode de configuration manuelle, vous devez spécifier manuellement les configurations du tunnel lors de la création du tunnel. Après avoir créé le tunnel, vous pouvez utiliser SSH dans le navigateur ou ouvrir un terminal en dehors de la AWS IoT console. Ce didacticiel explique comment utiliser le terminal situé en dehors de la console pour accéder à l'appareil distant. Vous allez également apprendre à configurer le proxy local, puis à vous connecter au proxy local pour interagir avec le périphérique distant. Pour vous connecter au proxy local, vous devez télécharger le jeton d'accès à la source lors de la création du tunnel.

Avec cette méthode de configuration, vous pouvez utiliser des services autres que SSH, tels que le FTP pour vous connecter au périphérique distant. Pour plus d’informations, veuillez consulter le AWSIoTWirelessFullAccess récapitulatif de la stratégie. Méthodes de configuration des tunnels.

Conditions préalables à la méthode de configuration manuelle

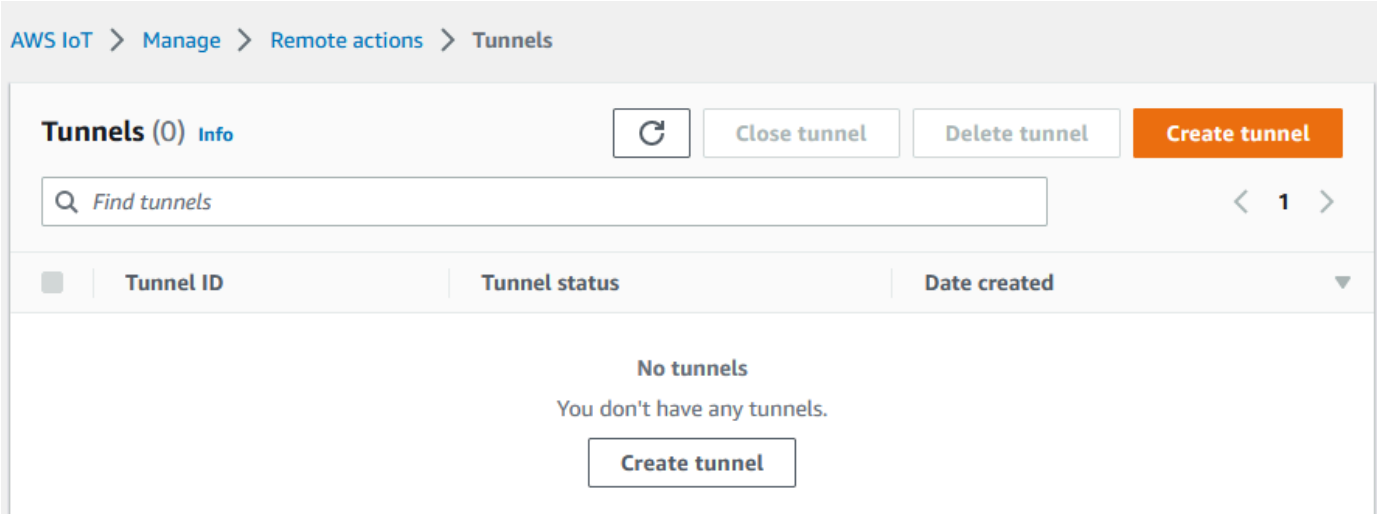
- Les pare-feu derrière l'appareil distant doivent permettre le trafic sortant sur le port 443. Le tunnel que vous créez utilisera ce port pour se connecter au périphérique distant.
- Vous disposez d'un agent pour appareils IoT (voirExtrait de l'agent IoT) exécuté sur l'appareil distant qui se connecte à la passerelle de l' AWS IoT appareil et qui est configuré avec un abonnement à une rubrique MQTT. Pour plus d'informations, voir connecter un appareil à la passerelle de AWS IoT périphériques.
- Vous devez disposer d'un démon SSH s'exécutant sur l'appareil distant.
- Vous avez téléchargé le code source du proxy local depuis GitHubet l'avez créé pour la plateforme de votre choix. Dans ce didacticiel, nous utilisons `localproxy` pour nous référer au fichier exécutable du proxy local.

Ouvrir un tunnel

Vous pouvez ouvrir un tunnel sécurisé à l'aide du AWS Management Console, de la référence d' AWS IoT API ou du AWS CLI. Vous pouvez éventuellement configurer un nom de destination, mais cela n'est pas obligatoire pour ce didacticiel. Si vous configurez la destination, le tunneling sécurisé fournira automatiquement le jeton d'accès au périphérique distant à l'aide de MQTT. Pour de plus amples informations, veuillez consulter Méthodes de création de tunnels dans AWS IoT la console.

Pour ouvrir un tunnel dans la console

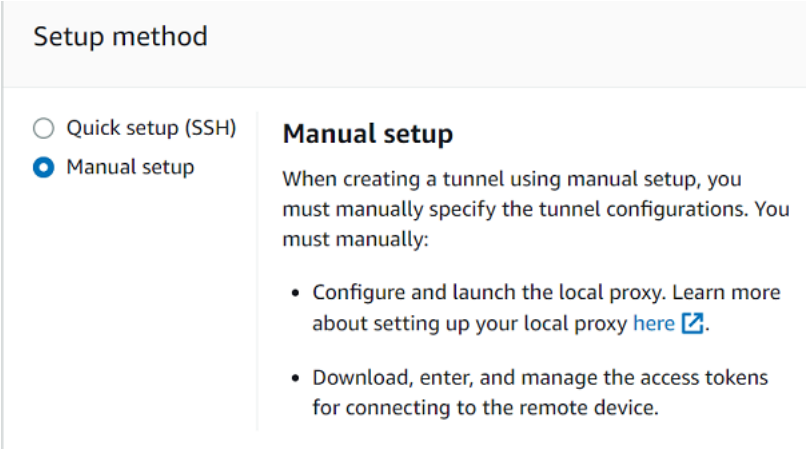
1. Accédez au hub Tunnels de la AWS IoT console et choisissez Create tunnel.



2. Pour ce didacticiel, choisissez Configuration manuelle comme méthode de création du tunnel, puis cliquez sur Suivant. Pour plus d'informations sur l'utilisation de la méthode de configuration

rapide pour créer un tunnel, consultezOuvrez un tunnel et utilisez le protocole SSH basé sur un navigateur pour accéder à un appareil distant.

> **Note**
>
> Si vous créez un tunnel sécurisé à partir de la page de détails d'un objet, vous pouvez choisir de créer un nouveau tunnel ou d'utiliser un tunnel existant. Pour de plus amples informations, veuillez consulter Ouvrez un tunnel pour un appareil distant et utilisez le protocole SSH basé sur un navigateur.



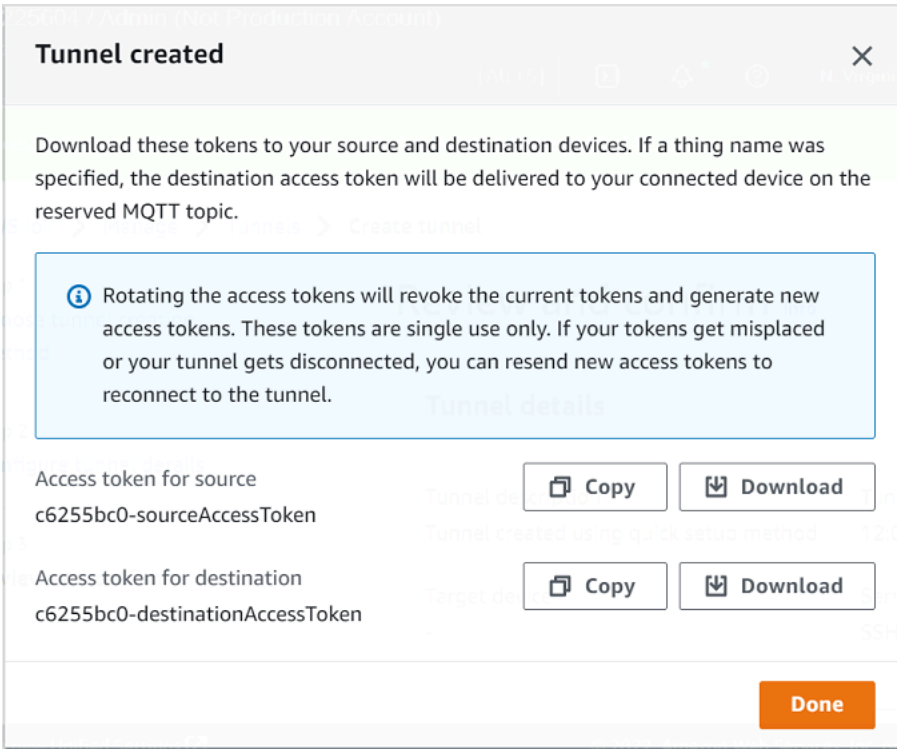
3. (Facultatif) Entrez les paramètres de configuration de votre tunnel. Vous pouvez également ignorer cette étape et passer à l'étape suivante pour créer un tunnel.

   Entrez une description du tunnel, une durée d'expiration du tunnel et des balises de ressource sous forme de paires clé-valeur pour vous aider à identifier votre ressource. Pour ce didacticiel, vous pouvez ignorer la configuration de la destination.

   > **Note**
   >
   > Aucun frais ne vous sera facturé en fonction de la durée pendant laquelle vous maintenez un tunnel ouvert. Vous ne payez des frais que lors de la création d'un nouveau tunnel. Pour plus d'informations sur les tarifs, voir Secure Tunneling dans AWS IoT Device Management la section des tarifs.

4. Téléchargez les jetons d'accès client, puis choisissez OK. Les jetons ne pourront pas être téléchargés une fois que vous aurez sélectionné OK.

Ces jetons ne peuvent être utilisés qu'une seule fois pour se connecter au tunnel. Si vous égarez les jetons ou si le tunnel est déconnecté, vous pouvez générer et envoyer de nouveaux jetons à votre appareil distant pour vous reconnecter au tunnel.



Pour ouvrir un tunnel à l'aide de l'API

Pour ouvrir un nouveau tunnel, vous pouvez utiliser l'opération OpenTunnelAPI. Vous pouvez également spécifier des configurations supplémentaires à l'aide de l'API, telles que la durée du tunnel et la configuration de destination.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

L'exécution de cette commande crée un nouveau tunnel et vous fournit les jetons d'accès à la source et à la destination.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
```

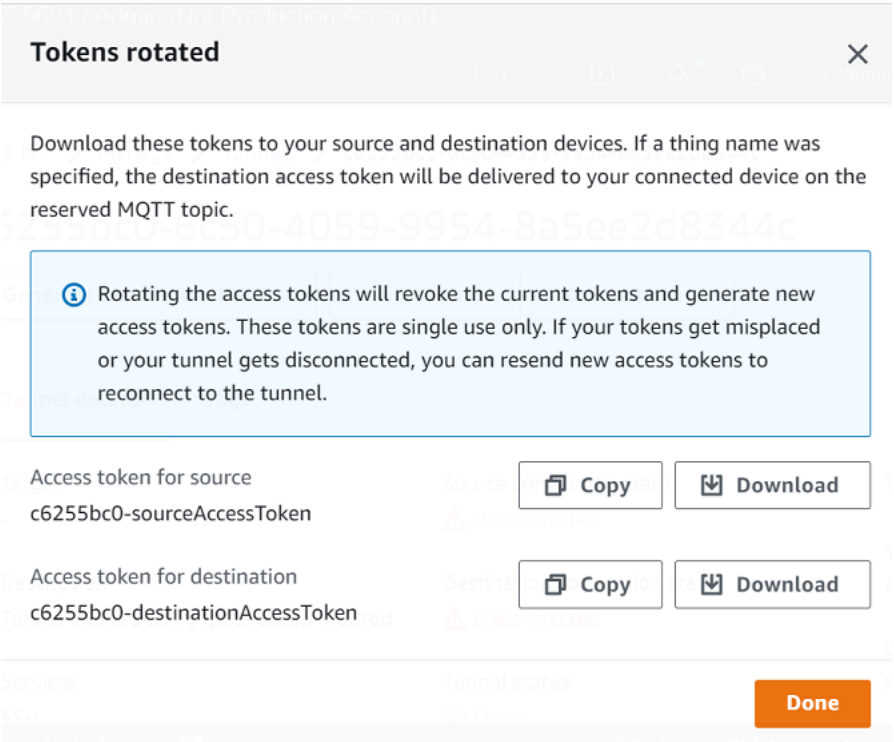
```
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Renvoyer les jetons d'accès au tunnel

Les jetons que vous avez obtenus lors de la création d'un tunnel ne peuvent être utilisés qu'une seule fois pour vous connecter au tunnel. Si vous égarez le jeton d'accès ou si le tunnel est déconnecté, vous pouvez renvoyer de nouveaux jetons d'accès à l'appareil distant en utilisant MQTT sans frais supplémentaires. AWS IoT le tunneling sécurisé révoquera les jetons actuels et renverra de nouveaux jetons d'accès pour la reconnexion au tunnel.

Pour faire pivoter les jetons depuis la console

1. Accédez au hub Tunnels de la AWS IoT console et choisissez le tunnel que vous avez créé.
2. Sur la page des détails du tunnel, choisissez Générer de nouveaux jetons d'accès, puis cliquez sur Suivant.
3. Téléchargez les nouveaux jetons d'accès pour votre tunnel et choisissez OK. Ces jetons ne peuvent être utilisés qu'une seule fois. Si vous égarez ces jetons ou si le tunnel est déconnecté, vous pouvez renvoyer de nouveaux jetons d'accès.



Pour faire pivoter les jetons d'accès à l'aide de l'API

Pour faire pivoter les jetons d'accès au tunnel, vous pouvez utiliser l'opération RotateTunnelAccessTokenAPI pour révoquer les jetons actuels et renvoyer de nouveaux jetons d'accès pour vous reconnecter au tunnel. Par exemple, la commande suivante fait pivoter les jetons d'accès pour le périphérique de destination, *RemoteThing1*.

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

L'exécution de cette commande génère le nouveau jeton d'accès, comme indiqué dans l'exemple suivant. Le jeton est ensuite envoyé à l'appareil à l'aide de MQTT pour se connecter au tunnel, si l'agent du périphérique est correctement configuré.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Pour des exemples montrant comment et quand faire pivoter les jetons d'accès, consultezRésolution des problèmes de connectivité liés au tunneling AWS IoT sécurisé en faisant pivoter les jetons d'accès client.

Configuration et démarrage du proxy local

Pour vous connecter à l'appareil distant, ouvrez un terminal sur votre ordinateur portable, puis configurez et démarrez le proxy local. Le proxy local transmet les données envoyées par l'application exécutée sur le périphérique source en utilisant un tunneling sécurisé via une connexion WebSocket sécurisée. Vous pouvez télécharger la source du proxy local depuis GitHub.

Après avoir configuré le proxy local, copiez le jeton d'accès du client source et utilisez-le pour démarrer le proxy local en mode source. Voici un exemple de commande pour démarrer le proxy local. Dans la commande suivante, le proxy local est configuré pour écouter les nouvelles connexions sur le port 5555. Dans cette commande :

- `-r`spécifie le Région AWS, qui doit être la même région que celle dans laquelle votre tunnel a été créé.

- `-s` Spécifie le port auquel le proxy doit se connecter.

- `-t` Spécifie le texte du jeton client.

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

L'exécution de cette commande démarrera le proxy local en mode source. Si vous recevez l'erreur suivante après avoir exécuté la commande, configurez le chemin d'accès à l'autorité de certification. Pour plus d'informations, voir Secure tunneling local proxy activé. GitHub

```
Could not perform SSL handshake with proxy server: certificate verify failed
```

Ce qui suit montre un exemple de sortie de l'exécution du proxy local en `source` mode.

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:
```

```
...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Démarrer une session SSH

Ouvrez un autre terminal et utilisez la commande suivante pour démarrer une nouvelle session SSH en vous connectant au proxy local sur le port 5555.

```
ssh username@localhost -p 5555
```

Vous pouvez être invité à entrer un mot de passe pour la session SSH. Lorsque vous avez terminé avec la session SSH, tapez **`exit`** pour fermer la session.

Nettoyage

- Fermer Tunnel

  Nous vous recommandons de fermer le tunnel une fois que vous avez fini de l'utiliser. Un tunnel peut également être fermé s'il est resté ouvert plus longtemps que la durée de tunnel spécifiée. Un tunnel ne peut pas être rouvert une fois fermé. Vous pouvez toujours dupliquer un tunnel en ouvrant le tunnel fermé, puis en choisissant Dupliquer le tunnel. Spécifiez la durée du tunnel que vous souhaitez utiliser, puis créez le nouveau tunnel.

  - Pour fermer un tunnel individuel ou plusieurs tunnels depuis la AWS IoT console, accédez au hub Tunnels, choisissez les tunnels que vous souhaitez fermer, puis cliquez sur Fermer le tunnel.
  - Pour fermer un tunnel individuel ou plusieurs tunnels à l'aide de l' AWS IoT API de référence API, utilisez l'opération CloseTunnelAPI.

    ```
    aws iotsecuretunneling close-tunnel \
        --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    ```

- Supprimer le tunnel

  Vous pouvez supprimer définitivement un tunnel de votre Compte AWS.

> **Warning**
>
> Les actions de suppression sont permanentes et ne peuvent être annulées.

- Pour supprimer un ou plusieurs tunnels de la AWS IoT console, accédez au hub Tunnels, choisissez les tunnels que vous souhaitez supprimer, puis sélectionnez Supprimer le tunnel.
- Pour supprimer un tunnel individuel ou plusieurs tunnels à l'aide de l' AWS IoT API de référence API, utilisez l'opération CloseTunnelAPI. Lorsque vous utilisez l'API, définissez l'`delete`indicateur sur`true`.

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Ouvrez un tunnel pour un appareil distant et utilisez le protocole SSH basé sur un navigateur

Depuis la AWS IoT console, vous pouvez créer un tunnel depuis le hub Tunnels ou depuis la page de détails d'un objet IoT que vous avez créé. Lorsque vous créez un tunnel à partir du hub Tunnels, vous pouvez spécifier si vous souhaitez créer un tunnel à l'aide de la configuration rapide ou de la configuration manuelle. Pour voir un exemple de didacticiel, consultez la section Ouvrez un tunnel et démarrez une session SSH sur un appareil distant.

Lorsque vous créez un tunnel à partir de la page de détails de l'objet de la AWS IoT console, vous pouvez également spécifier s'il faut créer un nouveau tunnel ou ouvrir un tunnel existant pour cet objet, comme illustré dans ce didacticiel. Si vous choisissez un tunnel existant, vous pouvez accéder au tunnel ouvert le plus récent que vous avez créé pour cet appareil. Vous pouvez ensuite utiliser l'interface de ligne de commande du terminal pour accéder à l'appareil par SSH.

Prérequis

- Les pare-feu derrière l'appareil distant doivent autoriser permettre le trafic sortant sur le port 443. Le tunnel que vous créez utilisera ce port pour se connecter au périphérique distant.

- Vous avez créé un objet IoT (par exemple,`RemoteDevice1`) dans le AWS IoT registre. Cela correspond à la représentation de votre appareil distant dans le cloud. Pour plus d'informations, consultez Enregistrement d'un appareil dans le AWS IoT registre.
- Vous disposez d'un agent pour appareils IoT (voirExtrait de l'agent IoT) exécuté sur l'appareil distant qui se connecte à la passerelle de l' AWS IoT appareil et qui est configuré avec un abonnement à une rubrique MQTT. Pour plus d'informations, voir connecter un appareil à la passerelle de AWS IoT périphériques.
- Vous devez disposer d'un démon SSH s'exécutant sur l'appareil distant.

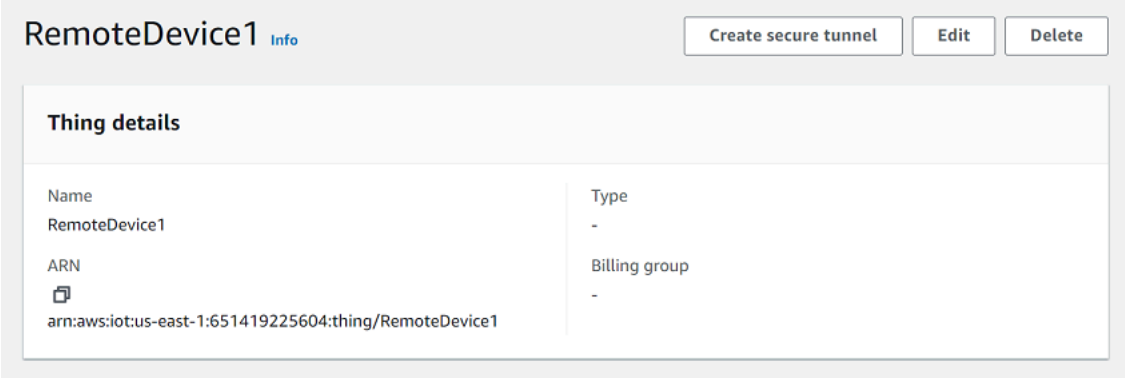
Ouvrez un nouveau tunnel pour le périphérique distant

Supposons que vous souhaitiez ouvrir un tunnel vers votre appareil distant, `RemoteDevice1`. Tout d'abord, créez un objet IoT dont le nom `RemoteDevice1` figure dans le AWS IoT registre. Vous pouvez ensuite créer un tunnel à l' AWS Management Console aide de l' AWS IoT API de référence ou du AWS CLI.

Si vous configurez la destination lorsque vous appelez , le service Secure Tunneling fournit le jeton d'accès du client de destination à l'appareil distant via MQTT et la rubrique MQTT réservée (`$aws/things/RemoteDeviceA/tunnels/notify`). Pour de plus amples informations, veuillez consulter Méthodes de création de tunnels dans AWS IoT la console.

Pour créer un tunnel pour un appareil distant à partir de la console

1. Choisissez l'objet, `RemoteDevice1`, pour afficher ses détails, puis choisissez Créer un tunnel sécurisé.



2. Choisissez de créer un nouveau tunnel ou d'ouvrir un tunnel existant. Pour créer un nouveau tunnel, choisissez Create new tunnel. Vous pouvez ensuite choisir d'utiliser la méthode de configuration manuelle ou la méthode de configuration rapide pour créer le tunnel. Pour plus d'informations, consultez Ouvrez un tunnel à l'aide de la configuration manuelle et connectez-

vous à un appareil distant et Ouvrez un tunnel et utilisez le protocole SSH basé sur un navigateur pour accéder à un appareil distant.

Pour créer un tunnel pour un appareil distant à l'aide de l'API

Pour ouvrir un nouveau tunnel, vous pouvez utiliser l'opération OpenTunnelAPI. Le code suivant montre un exemple d'exécution de cette commande.

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
    --cli-input-json file://input.json
```

L'exemple suivant affiche le contenu du fichier `input.json`. Vous pouvez utiliser le `destinationConfig` paramètre pour spécifier le nom de l'appareil de destination (par exemple,*RemoteDevice1*) et le service que vous souhaitez utiliser pour accéder à l'appareil de destination, par exemple*SSH*. En option, vous pouvez également spécifier des paramètres supplémentaires tels que la description du tunnel et les balises.

Contenu de input.json

```
{
   "description": "Tunnel to remote device1",
   "destinationConfig": {
      "services": [ "SSH" ],
      "thingName": "RemoteDevice1"
   }
}
```

L'exécution de cette commande crée un nouveau tunnel et vous fournit les jetons d'accès à la source et à la destination.

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

Ouvrez un tunnel existant et utilisez le protocole SSH basé sur un navigateur

Supposons que vous ayez créé le tunnel pour votre appareil distant à l'aide de la méthode de configuration manuelle ou de l' AWS IoT API de référence de l'API. `RemoteDevice1` Vous pouvez ensuite ouvrir le tunnel existant pour l'appareil et choisir Configuration rapide pour utiliser la fonctionnalité SSH basée sur un navigateur. Les configurations d'un tunnel existant ne peuvent pas être modifiées, vous ne pouvez donc pas utiliser la méthode de configuration manuelle.

Pour utiliser la fonctionnalité SSH basée sur un navigateur, vous n'avez pas besoin de télécharger le jeton d'accès à la source ni de configurer le proxy local. Un proxy local basé sur le Web sera automatiquement configuré pour vous afin que vous puissiez commencer à interagir avec votre appareil distant.

Pour utiliser la méthode de configuration rapide et le protocole SSH basé sur un navigateur

1. Accédez à la page de détails de l'objet que vous avez créé`RemoteDevice1`, puis créez un tunnel sécurisé.
2. Choisissez Utiliser un tunnel existant pour ouvrir le tunnel ouvert le plus récent que vous avez créé pour le périphérique distant. Les configurations du tunnel ne peuvent pas être modifiées, vous ne pouvez donc pas utiliser la méthode de configuration manuelle pour le tunnel. Pour utiliser la méthode de configuration rapide, choisissez Configuration rapide.
3. Passez en revue et confirmez les détails de configuration du tunnel et créez le tunnel. Les configurations des tunnels ne peuvent pas être modifiées.

   Lorsque vous créez le tunnel, le tunneling sécurisé utilise l'opération RotateTunnelAccessTokenAPI pour révoquer les jetons d'accès d'origine et générer de nouveaux jetons d'accès. Si votre appareil distant utilise le MQTT, ces jetons seront automatiquement envoyés à l'appareil distant sur le thème MQTT auquel il est abonné. Vous pouvez également choisir de télécharger ces jetons manuellement sur votre appareil source.

Après avoir créé le tunnel, vous pouvez utiliser le SSH basé sur un navigateur pour interagir avec le périphérique distant directement depuis la console à l'aide de l'interface de ligne de commande contextuelle. Pour utiliser cette interface de ligne de commande, choisissez le tunnel correspondant à l'objet que vous avez créé et, dans la page de détails, développez la section Interface de ligne de commande. Comme le proxy local a déjà été configuré pour vous, vous pouvez commencer à saisir des commandes pour commencer rapidement à accéder à votre appareil distant et à interagir avec celui-ci, `RemoteDevice1`.

Pour plus d'informations sur la méthode de configuration rapide et l'utilisation du SSH basé sur un navigateur, consultez Ouvrez un tunnel et utilisez le protocole SSH basé sur un navigateur pour accéder à un appareil distant.

Nettoyage

- Fermer Tunnel

  Nous vous recommandons de fermer le tunnel une fois que vous avez fini de l'utiliser. Un tunnel peut également être fermé s'il est resté ouvert plus longtemps que la durée de tunnel spécifiée. Un tunnel ne peut pas être rouvert une fois fermé. Vous pouvez toujours dupliquer un tunnel en ouvrant le tunnel fermé, puis en choisissant Dupliquer le tunnel. Spécifiez la durée du tunnel que vous souhaitez utiliser, puis créez le nouveau tunnel.

  - Pour fermer un tunnel individuel ou plusieurs tunnels depuis la AWS IoT console, accédez au hub Tunnels, choisissez les tunnels que vous souhaitez fermer, puis cliquez sur Fermer le tunnel.
  - Pour fermer un tunnel individuel ou plusieurs tunnels à l'aide de l' AWS IoT API de référence API, utilisez l'opération CloseTunnelAPI.

    ```
    aws iotsecuretunneling close-tunnel \
        --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    ```

- Supprimer le tunnel

  Vous pouvez supprimer définitivement un tunnel de votre Compte AWS.

  > **Warning**
  >
  > Les actions de suppression sont permanentes et ne peuvent être annulées.

  - Pour supprimer un ou plusieurs tunnels de la AWS IoT console, accédez au hub Tunnels, choisissez les tunnels que vous souhaitez supprimer, puis sélectionnez Supprimer le tunnel.
  - Pour supprimer un tunnel individuel ou plusieurs tunnels à l'aide de l' AWS IoT API de référence API, utilisez l'opération CloseTunnelAPI. Lorsque vous utilisez l'API, définissez l'`delete`indicateur sur`true`.

    ```
    aws iotsecuretunneling close-tunnel \
    ```

```
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

Proxy local

Le proxy local transmet les données envoyées par l'application exécutée sur le périphérique source en utilisant un tunneling sécurisé via une connexion WebSocket sécurisée. Vous pouvez télécharger la source du proxy local depuis GitHub.

Le proxy local peut s'exécuter en deux modes : `source` ou `destination`. En mode source, le proxy local s'exécute sur le même appareil ou réseau que l'application cliente qui initie la connexion TCP. En mode destination, le proxy local s'exécute sur l'appareil distant, avec l'application de destination. Un seul tunnel peut prendre en charge jusqu'à trois flux de données à la fois en utilisant le multiplexage par tunnel. Pour chaque flux de données, le tunneling sécurisé utilise plusieurs connexions TCP, ce qui réduit le risque de temporisation. Pour de plus amples informations, veuillez consulter Multiplexez les flux de données et utilisez des connexions TCP simultanées dans un tunnel sécurisé.

Comment utiliser le proxy local

Vous pouvez exécuter le proxy local sur les appareils source et de destination pour transmettre les données aux points de terminaison sécurisés du tunneling. Si vos appareils se trouvent dans un réseau utilisant un proxy Web, celui-ci peut intercepter les connexions avant de les rediriger vers Internet. Dans ce cas, vous devez configurer votre proxy local pour utiliser le proxy Web. Pour de plus amples informations, veuillez consulter Configuration du proxy local pour les appareils utilisant un proxy Web.

Flux de travail proxy local

Les étapes suivantes montrent comment le proxy local est exécuté sur les appareils source et de destination.

1. Connect un proxy local pour un tunneling sécurisé

   Tout d'abord, le proxy local doit établir une connexion pour sécuriser le tunnel. Lorsque vous démarrez le proxy local, utilisez les arguments suivants :

   - `-r`Argument permettant de spécifier l' Région AWS endroit dans lequel le tunnel est ouvert.

- `-t` L'argument pour passer le jeton d'accès du client source ou de destination renvoyé par la fonction `OpenTunnel`.

> **Note**
>
> Deux proxy locaux utilisant la même valeur de jeton d'accès client ne peuvent pas être connectés en même temps.

2. Exécuter des actions à la source ou à la destination

   Une fois la WebSocket connexion établie, le proxy local exécute des actions en mode source ou en mode destination, en fonction de sa configuration.

   Par défaut, le proxy local tente de se reconnecter au tunneling sécurisé si des erreurs se produisent ou si la WebSocket connexion est fermée de façon inattendue. input/output (I/O Cela provoque la fermeture de la connexion TCP. Si des erreurs de socket TCP se produisent, le proxy local envoie un message via le tunnel pour avertir l'autre partie de fermer sa connexion TCP. Par défaut, le proxy local utilise toujours la communication SSL.

3. Arreter le proxy local

   Après avoir utilisé le tunnel, il est prudent d'arrêter le processus de proxy local. Nous vous recommandons de fermer explicitement le tunnel en appelant `CloseTunnel`. Les clients du tunnel actifs peuvent ne pas être fermés juste après l'appel`CloseTunnel`.

Pour plus d'informations sur l'utilisation du AWS Management Console pour ouvrir un tunnel et démarrer une session SSH, consultezOuvrez un tunnel et démarrez une session SSH sur un appareil distant.

Meilleures pratiques en matière de procuration locale

Lorsque vous utilisez le proxy local, suivez ces bonnes pratiques :

- Évitez d'utiliser l'argument de proxy local `-t` pour transmettre un jeton d'accès. Nous vous recommandons d'utiliser la variable d'environnement `AWSIOT_TUNNEL_ACCESS_TOKEN` pour définir le jeton d'accès pour le proxy local.
- Exécutez l'exécutable du proxy local avec moindres privilèges dans le système d'exploitation ou l'environnement.
  - Évitez d'exécuter le proxy local en tant qu'administrateur sous Windows.

  - Évitez d'exécuter le proxy local en tant que racine sur Linux et macOS.
- Envisagez d'exécuter le proxy local sur des hôtes distincts, des conteneurs, des bacs à sable, une chroot jail ou un environnement virtualisé.
- Créez le proxy local avec les indicateurs de sécurité pertinents correspondants à votre chaîne d'outils.
- Sur les appareils dotés de plusieurs interfaces réseau, utilisez l'argument `-b` pour lier le socket TCP à l'interface réseau utilisée pour communiquer avec l'application de destination.

Exemple de commande et de sortie

Vous pouvez voir ci-dessous un exemple de commande que vous exécutez et le résultat correspondant. L'exemple montre comment le proxy local peut être configuré dans `source` les deux `destination` modes. Le proxy local met à niveau le protocole HTTPS WebSockets pour établir une connexion de longue durée, puis commence à transmettre des données via la connexion aux points de terminaison du dispositif de tunneling sécurisé.

Avant d'exécuter ces commandes :

Vous devez avoir ouvert un tunnel et obtenu les jetons d'accès client pour la source et la destination. Vous devez également avoir créé le proxy local comme décrit précédemment. Pour créer le proxy local, ouvrez le code source du proxy local dans le GitHub référentiel et suivez les instructions de création et d'installation du proxy local.

> **Note**
>
> Les commandes suivantes utilisées dans les exemples utilisent l'`verbosity`indicateur pour illustrer une vue d'ensemble des différentes étapes décrites précédemment après l'exécution du proxy local. Nous vous recommandons d'utiliser cet indicateur uniquement à des fins de test.

Exécution d'un proxy local en mode source

Les commandes suivantes indiquent comment exécuter le proxy local en mode source.

Linux/macOS

Sous Linux ou macOS, exécutez les commandes suivantes dans le terminal pour configurer et démarrer le proxy local sur votre source.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -s 5555 -v 5 -r us-west-2
```

Où :

- `-s` est le port d'écoute de la source, qui démarre le proxy local en mode source.
- `-v` est la verbosité de la sortie, qui peut être une valeur comprise entre zéro et six.
- `-r` est la région du point final où le tunnel est ouvert.

Pour plus d'informations sur les paramètres, voir Options définies à l'aide d'arguments de ligne de commande.

Windows

Sous Windows, vous configurez le proxy local de la même manière que pour Linux ou macOS, mais la façon dont vous définissez les variables d'environnement est différente de celle des autres plateformes. Exécutez les commandes suivantes dans la `cmd` fenêtre pour configurer et démarrer le proxy local sur votre source.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -s 5555 -v 5 -r us-west-2
```

Où :

- `-s` est le port d'écoute de la source, qui démarre le proxy local en mode source.
- `-v` est la verbosité de la sortie, qui peut être une valeur comprise entre zéro et six.
- `-r` est la région du point final où le tunnel est ouvert.

Pour plus d'informations sur les paramètres, voir Options définies à l'aide d'arguments de ligne de commande.

> **Note**
>
> Lorsque vous utilisez la dernière version du proxy local en mode source, vous devez inclure le AWS CLI paramètre `--destination-client-type V1` sur le périphérique source pour des raisons de rétrocompatibilité. Cela s'applique lors de la connexion à l'un de ces modes de destination :
>
> - AWS IoT Client de l'appareil
> - AWS IoT Composant de tunneling sécurisé ou composant de tunneling AWS IoT Greengrass Version 2 sécurisé
> - Tout code de démonstration de AWS IoT Secure Tunneling écrit avant 2022
> - Versions 1.X du proxy local
>
> Ce paramètre garantit une communication correcte entre le proxy source mis à jour et les anciens clients de destination. Pour plus d'informations sur les versions de proxy locales, voir AWS IoT Secure Tunneling activé. GitHub

Voici un exemple de sortie illustrant l'exécution du proxy local en `source` mode.

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...
```

```
Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Exécution d'un proxy local en mode destination

Les commandes suivantes indiquent comment exécuter le proxy local en mode destination.

Linux/macOS

Sous Linux ou macOS, exécutez les commandes suivantes dans le terminal pour configurer et démarrer le proxy local sur votre destination.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -d 22 -v 5 -r us-west-2
```

Où :

- `-d` est l'application de destination qui démarre le proxy local en mode destination.
- `-v` est la verbosité de la sortie, qui peut être une valeur comprise entre zéro et six.
- `-r` est la région du point final où le tunnel est ouvert.

Pour plus d'informations sur les paramètres, voir Options définies à l'aide d'arguments de ligne de commande.

Windows

Sous Windows, vous configurez le proxy local de la même manière que pour Linux ou macOS, mais la façon dont vous définissez les variables d'environnement est différente de celle des autres plateformes. Exécutez les commandes suivantes dans la `cmd` fenêtre pour configurer et démarrer le proxy local sur votre destination.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -d 22 -v 5 -r us-west-2
```

Où :

- `-d` est l'application de destination qui démarre le proxy local en mode destination.
- `-v` est la verbosité de la sortie, qui peut être une valeur comprise entre zéro et six.
- `-r` est la région du point final où le tunnel est ouvert.

Pour plus d'informations sur les paramètres, voir Options définies à l'aide d'arguments de ligne de commande.

> **Note**
>
> Lorsque vous utilisez la dernière version du proxy local en mode destination, vous devez inclure le AWS CLI paramètre `--destination-client-type V1` sur le périphérique de destination pour des raisons de rétrocompatibilité. Cela s'applique lors de la connexion à l'un de ces modes source :
>
> - Tunneling sécurisé basé sur un navigateur depuis la console. AWS
> - Versions 1.X du proxy local
>
> Ce paramètre garantit une communication correcte entre le proxy de destination mis à jour et les anciens clients source. Pour plus d'informations sur les versions de proxy locales, voir AWS IoT Secure Tunneling activé. GitHub

Voici un exemple de sortie illustrant l'exécution du proxy local en `destination` mode.

```
...
...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
```

```
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

Configuration du proxy local pour les appareils utilisant un proxy Web

Vous pouvez utiliser un proxy local sur les AWS IoT appareils pour communiquer grâce à un tunneling AWS IoT APIs sécurisé. Le proxy local transmet les données envoyées par l'application de l'appareil à l'aide d'un tunneling sécurisé via une connexion WebSocket sécurisée. Le proxy local peut fonctionner en mode `destination` ou `source`. En `source` mode, il s'exécute sur le même appareil ou le même réseau que celui qui initie la connexion TCP. En `destination` mode, le proxy local s'exécute sur l'appareil distant, en même temps que l'application de destination. Pour de plus amples informations, veuillez consulter Proxy local.

Le proxy local doit se connecter directement à Internet pour utiliser le AWS IoT tunneling sécurisé. Pour une connexion TCP de longue durée avec tunneling sécurisé, le proxy local met à niveau la requête HTTPS pour établir une WebSockets connexion à l'un des points de terminaison de connexion du dispositif de tunneling sécurisé.

Si vos appareils se trouvent dans un réseau qui utilise un proxy Web, celui-ci peut intercepter les connexions avant de les rediriger vers Internet. Pour établir une connexion de longue durée avec les

points de terminaison de connexion du dispositif de tunneling sécurisé, configurez votre proxy local pour qu'il utilise le proxy Web comme décrit dans la spécification du websocket.

> **Note**
>
> Le AWS IoT Client de l'appareil ne prend pas en charge les appareils qui utilisent un proxy web. Pour utiliser le proxy Web, vous devez utiliser un proxy local et le configurer pour qu'il fonctionne avec un proxy Web, comme décrit ci-dessous.

Les étapes suivantes montrent comment le proxy local fonctionne avec un proxy Web.

1. Le proxy local envoie une `CONNECT` requête HTTP au proxy Web qui contient l'adresse distante du service de tunneling sécurisé, ainsi que les informations d'authentification du proxy Web.
2. Le proxy Web créera ensuite une connexion de longue durée avec les points de terminaison de tunneling sécurisés distants.
3. La connexion TCP est établie et le proxy local fonctionne désormais en mode source et en mode destination pour la transmission de données.

Pour mener à bien cette procédure, procédez comme suit.

- Créez le proxy local
- Configuration de votre proxy Web
- Configuration et démarrage du proxy local

Créez le proxy local

Ouvrez le code source du proxy local dans le GitHub référentiel et suivez les instructions de création et d'installation du proxy local.

Configuration de votre proxy Web

Le proxy local repose sur le mécanisme de tunneling HTTP décrit par la spécification HTTP/1.1. Pour être conforme aux spécifications, votre proxy Web doit autoriser les appareils à utiliser la `CONNECT` méthode.

La façon dont vous configurez votre proxy Web dépend du proxy Web que vous utilisez et de la version du proxy Web. Pour vous assurer que vous configurez correctement le proxy Web, consultez la documentation de votre proxy Web.

Pour configurer votre proxy Web, identifiez d'abord l'URL de votre proxy Web et vérifiez si celui-ci prend en charge le tunneling HTTP. L'URL du proxy Web sera utilisée ultérieurement lorsque vous configurerez et démarrerez le proxy local.

1. Identifiez l'URL de votre proxy Web

   Le format de votre URL de proxy web sera au format suivant.

   ```
   protocol://web_proxy_host_domain:web_proxy_port
   ```

   AWS IoT le tunneling sécurisé ne prend en charge que l'authentification de base pour le proxy Web. Pour utiliser l'authentification de base, vous devez spécifier le **`username`** et dans le **`password`** cadre de l'URL du proxy Web. L'URL du proxy web aura le format suivant.

   ```
   protocol://username:password@web_proxy_host_domain:web_proxy_port
   ```

   - *`protocol`*peut être `http` ou`https`. Nous vous recommandons d'utiliser `https`.
   - *`web_proxy_host_domain`*est l'adresse IP de votre proxy Web ou un nom DNS correspondant à l'adresse IP de votre proxy Web.
   - *`web_proxy_port`*est le port sur lequel le proxy Web écoute.
   - Le proxy Web l'utilise **`username`** et **`password`** pour authentifier la demande.
2. Testez l'URL de votre proxy web

   Pour vérifier si votre proxy Web prend en charge le tunneling TCP, utilisez une `curl` commande et assurez-vous d'obtenir une réponse réponse `2xx` ou une `3xx`.

   Par exemple, si l'URL de votre proxy Web est`https://server.com:1235`, utilisez un `proxy-insecure` indicateur avec la `curl` commande, car le proxy Web peut s'appuyer sur un certificat auto-signé.

   ```
   export HTTPS_PROXY=https://server.com:1235
   curl -I https://aws.amazon.com --proxy-insecure
   ```

Si l'URL de votre proxy Web possède un `http` port (par exemple,`http://server.com:1234`), vous n'êtes pas obligé d'utiliser l'`proxy-insecure`indicateur.

```
export HTTPS_PROXY=http://server.com:1234
curl -I https://aws.amazon.com
```

Configuration et démarrage du proxy local

Pour configurer le proxy local afin qu'il utilise un proxy web, vous devez configurer la variable `HTTPS_PROXY` d'environnement avec les noms de domaine DNS ou les adresses IP et les numéros de port utilisés par votre proxy web.

Après avoir configuré le proxy local, vous pouvez utiliser le proxy local comme expliqué dans ce document README.

> **Note**
>
> La casse est sensible à la casse dans votre déclaration de variable d'environnement. Nous vous recommandons de définir chaque variable une seule fois en majuscules ou en minuscules. Les exemples suivants montrent que la variable d'environnement est déclarée en lettres majuscules. Si la même variable est spécifiée à la fois en majuscules et en minuscules, la variable spécifiée en minuscules est prioritaire.

Les commandes suivantes indiquent comment configurer le proxy local qui s'exécute sur votre destination pour utiliser le proxy Web et démarrer le proxy local.

- `AWSIOT_TUNNEL_ACCESS_TOKEN` : Cette variable contient le jeton d'accès client (CAT) pour la destination.
- `HTTPS_PROXY` : Cette variable contient l'URL du proxy Web ou l'adresse IP permettant de configurer le proxy local.

Les commandes présentées dans les exemples suivants dépendent du système d'exploitation que vous utilisez et du fait que le proxy Web écoute sur un port HTTP ou HTTPS.

Proxy Web écoutant sur un port HTTP

Si votre proxy Web écoute sur un port HTTP, vous pouvez fournir l'URL ou l'adresse IP du proxy Web pour la HTTPS_PROXY variable.

Linux/macOS

Sous Linux ou macOS, exécutez les commandes suivantes dans le terminal pour configurer et démarrer le proxy local sur votre destination afin d'utiliser un proxy Web écoutant un port HTTP.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Si vous devez vous authentifier auprès du proxy, vous devez spécifier un **username** et dans le **password** cadre de la HTTPS_PROXY variable.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

Windows

Sous Windows, vous configurez le proxy local de la même manière que pour Linux ou macOS, mais la façon dont vous définissez les variables d'environnement est différente de celle des autres plateformes. Exécutez les commandes suivantes dans la cmd fenêtre pour configurer et démarrer le proxy local sur votre destination afin d'utiliser un proxy Web écoutant un port HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22
```

Si vous devez vous authentifier auprès du proxy, vous devez spécifier un **username** et **password** dans le cadre de la HTTPS_PROXY variable.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22
```

Proxy Web écoutant sur un port HTTPS

Exécutez les commandes suivantes si votre proxy Web écoute sur un port HTTPS.

> **Note**
>
> Si vous utilisez un certificat auto-signé pour le proxy Web ou si vous exécutez le proxy local sur un système d'exploitation qui ne prend pas en charge OpenSSL en mode natif et ne dispose pas de configurations par défaut, vous devrez configurer vos certificats de proxy Web comme décrit dans la section Configuration des certificats du référentiel. GitHub

Les commandes suivantes ressembleront à la façon dont vous avez configuré votre proxy Web pour un proxy HTTP, à l'exception du fait que vous spécifierez également le chemin d'accès aux fichiers de certificats que vous avez installés, comme décrit précédemment.

Linux/macOS

Sous Linux ou macOS, exécutez les commandes suivantes dans le terminal pour configurer le proxy local exécuté sur votre destination afin qu'il utilise un proxy Web écoutant un port HTTPS.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Si vous devez vous authentifier auprès du proxy, vous devez spécifier un **`username`** et **`password`** dans le cadre de la HTTPS_PROXY variable.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Windows

Sous Windows, exécutez les commandes suivantes dans la cmd fenêtre pour configurer et démarrer le proxy local exécuté sur votre destination afin d'utiliser un proxy Web écoutant un port HTTP.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
```

```
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Si vous devez vous authentifier auprès du proxy, vous devez spécifier un **`username`** et **`password`** dans le cadre de la HTTPS_PROXY variable.

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

Exemple de commande et de sortie

Voici un exemple de commande que vous exécutez sur un système d'exploitation Linux et le résultat correspondant. L'exemple montre un proxy Web qui écoute sur un port HTTP et montre comment le proxy local peut être configuré pour utiliser le proxy Web dans modes `source` et `destination`. Avant de pouvoir exécuter ces commandes, vous devez avoir déjà ouvert un tunnel et obtenu les jetons d'accès client pour la source et la destination. Vous devez également avoir créé le proxy local et configuré votre proxy Web comme décrit précédemment.

Voici une vue d'ensemble des étapes à suivre après le démarrage du proxy local. Le proxy local :

- Identifie l'URL du proxy Web afin qu'il puisse l'utiliser pour se connecter au serveur proxy.
- Établit une connexion TCP avec le proxy Web.
- Envoie une `CONNECT` requête HTTP au proxy Web et attend la `HTTP/1.1 200` réponse, qui indique que la connexion a été établie.
- Met à niveau le protocole HTTPS WebSockets pour établir une connexion de longue durée.
- Commence à transmettre des données via la connexion aux points de terminaison du dispositif de tunneling sécurisé.

> **Note**
>
> Les commandes suivantes utilisées dans les exemples utilisent l'`verbosity`indicateur pour illustrer une vue d'ensemble des différentes étapes décrites précédemment après l'exécution du proxy local. Nous vous recommandons d'utiliser cet indicateur uniquement à des fins de test.

Exécution d'un proxy local en mode source

Les commandes suivantes montrent comment exécuter le proxy local en mode source.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -s 5555 -v 5 -r us-west-2
```

Ce qui suit montre un exemple de sortie de l'exécution du proxy local en `source` mode.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.

...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.11
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 0a109afffee745f5-00001341-000b8138-cc6c878d80e8adb0-f186064b
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

Exécution d'un proxy local en mode destination

Les commandes suivantes montrent comment exécuter le proxy local en mode destination.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -d 22 -v 5 -r us-west-2
```

Ce qui suit montre un exemple de sortie de l'exécution du proxy local en `destination` mode.

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via
 the proxy.

...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.1
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 06717bfffed3fd05-00001355-000b8315-da3109a85da804dd-24c3d10d
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

Multiplexez les flux de données et utilisez des connexions TCP simultanées dans un tunnel sécurisé

Vous pouvez utiliser plusieurs flux de données par tunnel en utilisant la fonction de multiplexage par tunneling sécurisé. Le multiplexage vous permet de dépanner des appareils utilisant plusieurs flux

de données. Vous pouvez également réduire votre charge opérationnelle en éliminant le besoin de créer, de déployer et de démarrer plusieurs proxys locaux ou d'ouvrir plusieurs tunnels vers le même appareil. Par exemple, le multiplexage peut être utilisé dans le cas d'un navigateur Web qui nécessite l'envoi de plusieurs flux de données HTTP et SSH.

Pour chaque flux de données, le tunneling AWS IoT sécurisé prend en charge les connexions TCP simultanées. L'utilisation de connexions simultanées réduit le risque de temporisation en cas de demandes multiples du client. Par exemple, cela peut réduire le temps de chargement lors de l'accès à distance à un serveur Web local à l'appareil de destination.

Les sections suivantes expliquent plus en détail le multiplexage et l'utilisation de connexions TCP simultanées, ainsi que leurs différents cas d'utilisation.

Rubriques

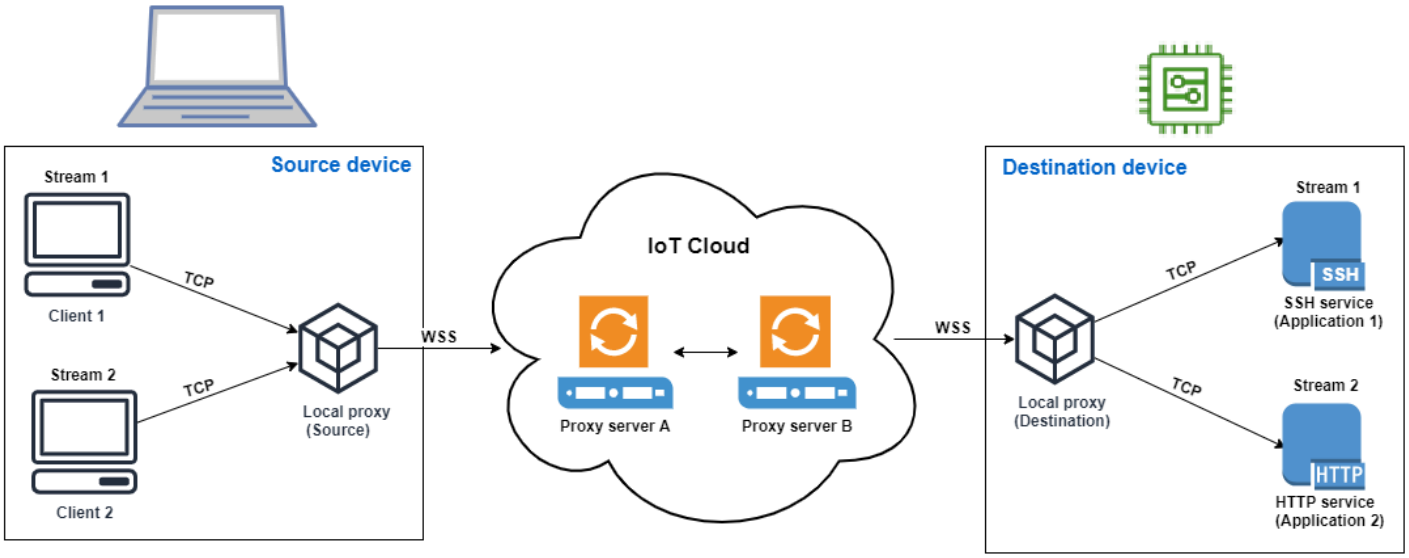
- Multiplexage de plusieurs flux de données dans un tunnel sécurisé
- Utilisation de connexions TCP simultanées dans un tunnel sécurisé

Multiplexage de plusieurs flux de données dans un tunnel sécurisé

Vous pouvez utiliser la fonction de multiplexage pour les appareils utilisant plusieurs connexions ou ports. Le multiplexage peut également être utilisé lorsque vous avez besoin de plusieurs connexions à un appareil distant pour résoudre des problèmes. Par exemple, il peut être utilisé dans le cas d'un navigateur Web qui nécessite l'envoi de plusieurs flux de données HTTP et SSH. Les données d'application provenant des deux flux sont envoyées au dispositif simultanément via le tunnel multiplexé.

Exemple de cas d’utilisation

Supposons que vous deviez vous connecter à une application Web intégrée à l'appareil pour modifier certains paramètres réseau, tout en émettant simultanément des commandes shell via le terminal pour vérifier que le périphérique fonctionne correctement avec les nouveaux paramètres réseau. Dans ce scénario, vous devrez peut-être vous connecter à l'appareil via HTTP et SSH et transférer deux flux de données parallèles pour accéder simultanément à l'application Web et au terminal. Grâce à la fonction de multiplexage, ces deux flux indépendants peuvent être transférés simultanément sur le même tunnel.



Configuration d'un tunnel multiplexé

La procédure suivante explique comment configurer un tunnel multiplexé pour le dépannage des périphériques à l'aide d'applications nécessitant des connexions à plusieurs ports. Vous allez configurer un tunnel avec deux flux multiplexés : un flux HTTP et un flux SSH.

1. (Facultatif) Créez des fichiers de configuration

   Vous pouvez éventuellement configurer le périphérique source et de destination à l'aide de fichiers de configuration. Utilisez des fichiers de configuration si vos mappages de ports sont susceptibles de changer fréquemment. Vous pouvez ignorer cette étape si vous préférez spécifier le mappage des ports de manière explicite à l'aide de la CLI, ou si vous n'avez pas besoin de démarrer le proxy local sur les ports d'écoute désignés. Pour plus d'informations sur l'utilisation des fichiers de configuration, voir Options définies via --config dans GitHub.

   1. Sur votre appareil source, dans le dossier où votre proxy local sera exécuté, créez un dossier de configuration appelé `Config`. Dans ce dossier, créez un fichier appelé `SSHSource.ini` avec le contenu suivant :

      ```
      HTTP1 = 5555
      SSH1 = 3333
      ```

   2. Sur votre appareil de destination, dans le dossier où votre proxy local sera exécuté, créez un dossier de configuration appelé`Config`. Dans ce dossier, créez un fichier appelé `SSHDestination.ini` avec le contenu suivant :

```
HTTP1 = 80
SSH1 = 22
```

2. Ouvrir un tunnel

   Ouvrez un tunnel à l'aide de l'opération `OpenTunnel` API ou de la commande `open-tunnel` CLI. Configurez la destination en spécifiant `SSH1` et en `HTTP1` tant que services et le nom de l' AWS IoT objet correspondant à votre appareil distant. Vos applications SSH et HTTP s'exécutent sur cet appareil distant. Vous devez déjà avoir créé l'objet IoT dans le AWS IoT registre. Pour de plus amples informations, veuillez consulter Gérer les choses avec le registre.

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP1,SSH1
   ```

   L'exécution de cette commande génère les jetons d'accès à la source et à la destination que vous utiliserez pour exécuter le proxy local.

   ```
   {
    "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
    "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
   afa28b00cecd",
    "sourceAccessToken": source_client_access_token,
    "destinationAccessToken": destination_client_access_token
   }
   ```

3. Configuration et démarrage du proxy local

   Avant de pouvoir exécuter le proxy local, configurez le client du AWS IoT périphérique ou téléchargez le code source du proxy local GitHubet créez-le pour la plate-forme de votre choix. Vous pouvez ensuite démarrer le proxy local de destination et de source pour vous connecter au tunnel sécurisé. Pour plus d'informations sur la configuration et l'utilisation du proxy local, voir Comment utiliser le proxy local.

   > **Note**
   >
   > Sur votre périphérique source, si vous n'utilisez aucun fichier de configuration ou si vous ne spécifiez pas le mappage des ports à l'aide de la CLI, vous pouvez toujours utiliser la même commande pour exécuter le proxy local. Le proxy local en mode source

sélectionnera automatiquement les ports disponibles à utiliser et les mappages pour vous.

Start local proxy using configuration files

Exécutez les commandes suivantes pour exécuter le proxy local dans les modes source et destination à l'aide de fichiers de configuration.

```
// ----------------- Start the destination local proxy ----------------------
./localproxy -r us-east-1 -m dst -t destination_client_access_token

// ----------------- Start the source local proxy ---------------------------
// You also run the same command below if you want the local proxy to
// choose the mappings for you instead of using configuration files.
./localproxy -r us-east-1 -m src -t source_client_access_token
```

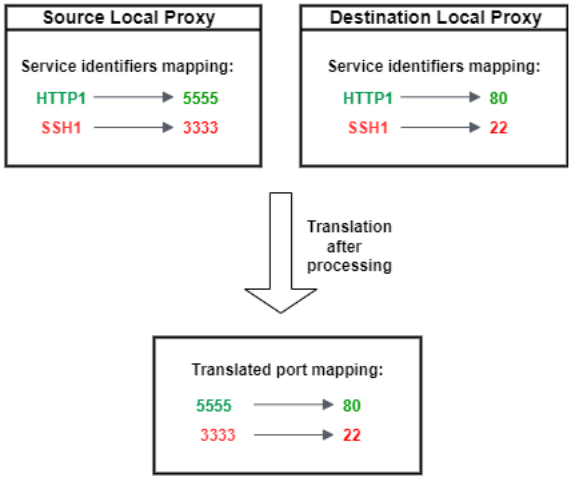
Start local proxy using CLI port mapping

Exécutez les commandes suivantes pour exécuter le proxy local dans les modes source et destination en spécifiant explicitement les mappages de ports à l'aide de la CLI.

```
// ----------------- Start the destination local proxy
 ----------------------------------
./localproxy -r us-east-1 -d HTTP1=80,SSH1=22 -t destination_client_access_token

// ----------------- Start the source local proxy
 ---------------------------------------
./localproxy -r us-east-1 -s HTTP1=5555,SSH1=33 -t source_client_access_token
```

Les données d'application issues des connexions SSH et HTTP peuvent désormais être transférées simultanément via le tunnel multiplexé. Comme le montre la carte ci-dessous, l'identifiant de service agit comme un format lisible pour traduire le mappage des ports entre le périphérique source et le périphérique de destination. Avec cette configuration, le tunneling sécurisé transmet tout trafic HTTP entrant du port du périphérique source *5555* au port *80* du périphérique de destination, et tout trafic SSH entrant d'un port *3333* à l'autre *22* sur le périphérique de destination.



Utilisation de connexions TCP simultanées dans un tunnel sécurisé

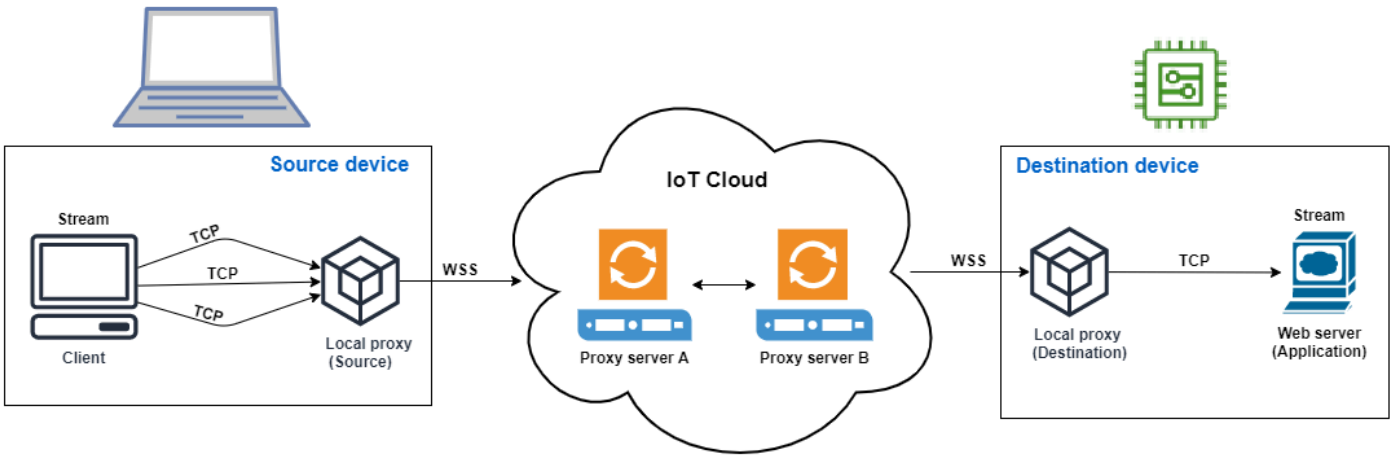
AWS IoT le tunneling sécurisé prend en charge plusieurs connexions TCP simultanément pour chaque flux de données. Vous pouvez utiliser cette fonctionnalité lorsque vous avez besoin de connexions simultanées à un appareil distant. L'utilisation de connexions TCP simultanées réduit le risque de temporisation en cas de demandes multiples du client. Par exemple, lorsque vous accédez à un serveur Web sur lequel plusieurs composants sont exécutés, les connexions TCP simultanées peuvent réduire le temps nécessaire au chargement du site.

> **Note**
>
> Les connexions TCP simultanées ont une limite de bande passante de 800 kilo-octets par seconde pour chacune. Compte AWS AWS IoT Secure Tunneling peut configurer cette limite pour vous en fonction du nombre de demandes entrantes.

Exemple de cas d’utilisation

Supposons que vous deviez accéder à distance à un serveur Web local sur l'appareil de destination et sur lequel plusieurs composants sont exécutés. Avec une seule connexion TCP, lorsque vous essayez d'accéder au serveur Web, le chargement séquentiel peut augmenter le temps nécessaire au chargement des ressources sur le site. Les connexions TCP simultanées peuvent réduire le temps de chargement en répondant aux besoins en ressources du site, réduisant ainsi le temps d'accès. Le schéma suivant montre comment les connexions TCP simultanées sont prises en charge pour le flux de données vers l'application de serveur Web exécutée sur le périphérique distant.

> **Note**
>
> Si vous souhaitez accéder à plusieurs applications exécutées sur le périphérique distant à l'aide du tunnel, vous pouvez utiliser le multiplexage par tunnel. Pour de plus amples informations, veuillez consulter Multiplexage de plusieurs flux de données dans un tunnel sécurisé.



Comment utiliser les connexions TCP simultanées

La procédure suivante explique comment utiliser des connexions TCP simultanées pour accéder au navigateur Web de l'appareil distant. Lorsque le client reçoit plusieurs demandes, le tunneling AWS IoT sécurisé configure automatiquement des connexions TCP simultanées pour traiter les demandes, réduisant ainsi le temps de chargement.

1. Ouvrir un tunnel

   Ouvrez un tunnel à l'aide de l'opération `OpenTunnel` API ou de la commande `open-tunnel` CLI. Configurez la destination `HTTP` en spécifiant le service et le nom de l' AWS IoT objet correspondant à votre appareil distant. Votre application de serveur Web est en cours d'exécution sur cet appareil distant. Vous devez déjà avoir créé l'objet IoT dans le AWS IoT registre. Pour de plus amples informations, veuillez consulter Gérer les choses avec le registre.

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP
   ```

L'exécution de cette commande génère les jetons d'accès à la source et à la destination que vous utiliserez pour exécuter le proxy local.

```
{
 "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
 "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
afa28b00cecd",
 "sourceAccessToken": source_client_access_token,
 "destinationAccessToken": destination_client_access_token
}
```

2. Configuration et démarrage du proxy local

Avant de pouvoir exécuter le proxy local, téléchargez le code source du proxy local GitHubet créez-le pour la plate-forme de votre choix. Vous pouvez ensuite démarrer le proxy local de destination et de source pour vous connecter au tunnel sécurisé et commencer à utiliser l'application de serveur Web distant.

> **Note**
>
> Pour que le tunneling AWS IoT sécurisé utilise des connexions TCP simultanées, vous devez passer à la dernière version du proxy local. Cette fonctionnalité n'est pas disponible si vous configurez le proxy local à l'aide du AWS IoT Device Client.

```
// Start the destination local proxy
./localproxy -r us-east-1 -d HTTP=80 -t destination_client_access_token

// Start the source local proxy
./localproxy -r us-east-1 -s HTTP=5555 -t source_client_access_token
```

Pour plus d'informations sur la configuration et l'utilisation du proxy local, voir Comment utiliser le proxy local.

Vous pouvez désormais utiliser le tunnel pour accéder à l'application du serveur Web. AWS IoT le tunneling sécurisé configurera et gérera automatiquement les connexions TCP simultanées en cas de demandes multiples du client.

Configuration d'un appareil distant et utilisation de l'agent IoT

L'agent IoT est utilisé pour recevoir le message MQTT qui inclut le jeton d'accès client et pour démarrer un proxy local sur l'appareil distant. Vous devez installer et exécuter l'agent IoT sur l'appareil distant si vous voulez que le tunnel sécurisé fournisse le jeton d'accès au client à l'aide de MQTT. L'agent IoT doit s'abonner à la rubrique MQTT IoT réservée suivante :

> **Note**
>
> Si vous souhaitez fournir le jeton d'accès au client de destination à l'appareil distant par des méthodes autres que l'abonnement à la rubrique MQTT réservée, vous aurez peut-être besoin d'un écouteur de jeton d'accès client (CAT) de destination et d'un proxy local. L'écouteur CAT doit fonctionner avec le mécanisme de délivrance de jetons d'accès client que vous avez choisi et être en mesure de démarrer un proxy local en mode destination.

Extrait de l'agent IoT

L'agent IoT doit s'abonner à la rubrique MQTT IoT réservée suivante afin de pouvoir recevoir le message MQTT et démarrer le proxy local :

`$aws/things/thing-name/tunnels/notify`

Où se `thing-name` trouve le nom de l' AWS IoT objet associé à l'appareil distant.

Voici un exemple de charge utile de message MQTT :

```
{
    "clientAccessToken": "destination-client-access-token",
    "clientMode": "destination",
    "region": "aws-region",
    "services": ["destination-service"]
}
```

Après avoir reçu un message MQTT, l'agent IoT doit démarrer un proxy local sur l'appareil distant avec les paramètres appropriés.

Le code Java suivant montre comment utiliser le SDK AWS IoT Device et ProcessBuilderla bibliothèque Java pour créer un agent IoT simple capable de fonctionner avec un tunneling sécurisé.

```
// Find the IoT device endpoint for your Compte AWS
final String endpoint = iotClient.describeEndpoint(new
 DescribeEndpointRequest().withEndpointType("iot:Data-ATS")).getEndpointAddress();

// Instantiate the IoT Agent with your AWS credentials
final String thingName = "RemoteDeviceA";
final String tunnelNotificationTopic = String.format("$aws/things/%s/tunnels/notify",
 thingName);
final AWSIotMqttClient mqttClient = new AWSIotMqttClient(endpoint, thingName,
                 "your_aws_access_key", "your_aws_secret_key");

try {
    mqttClient.connect();
    final TunnelNotificationListener listener = new
 TunnelNotificationListener(tunnelNotificationTopic);
    mqttClient.subscribe(listener, true);
}
finally {
    mqttClient.disconnect();
}

private static class TunnelNotificationListener extends AWSIotTopic {
    public TunnelNotificationListener(String topic) {
        super(topic);
    }

    @Override
    public void onMessage(AWSIotMessage message) {
        try {
            // Deserialize the MQTT message
            final JSONObject json = new JSONObject(message.getStringPayload());

            final String accessToken = json.getString("clientAccessToken");
            final String region = json.getString("region");

            final String clientMode = json.getString("clientMode");
            if (!clientMode.equals("destination")) {
                throw new RuntimeException("Client mode " + clientMode + " in the MQTT
 message is not expected");
            }

            final JSONArray servicesArray = json.getJSONArray("services");
            if (servicesArray.length() > 1) {
```

```
                throw new RuntimeException("Services in the MQTT message has more than
 1 service");
            }
            final String service = servicesArray.get(0).toString();
            if (!service.equals("SSH")) {
                throw new RuntimeException("Service " + service + " is not supported");
            }

            // Start the destination local proxy in a separate process to connect to
 the SSH Daemon listening port 22
            final ProcessBuilder pb = new ProcessBuilder("localproxy",
                        "-t", accessToken,
                        "-r", region,
                        "-d", "localhost:22");
            pb.start();
        }
        catch (Exception e) {
            log.error("Failed to start the local proxy", e);
        }
    }
}
```

Contrôle de l'accès aux tunnels

Le tunneling sécurisé fournit des actions, des ressources et des clés de contexte de condition spécifiques au service, à utiliser dans les politiques de permissions IAM.

Conditions préalables à l'accès au tunnel

- Découvrez comment sécuriser les AWS ressources à l'aide des politiques IAM.
- Découvrez comment créer et évaluer des conditions IAM.
- Découvrez comment sécuriser les AWS ressources à l'aide de balises de ressources.

Politiques d'accès aux tunnels

Vous devez utiliser les politiques suivantes pour autoriser les autorisations d'utilisation de l'API de tunneling sécurisé. Pour plus d'informations sur AWS IoT la sécurité, voirGestion des identités et des accès pour AWS IoT.

IoT : OpenTunnel

L'action de stratégie `iot:OpenTunnel` accorde à un mandataire l'autorisation d'appeler OpenTunnel.

Dans l'`Resource`élément de la déclaration de politique IAM :

- Spécifiez l'ARN du tunnel générique :

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`
- Spécifiez un objet ARN pour gérer les `OpenTunnel` autorisations pour des objets IoT spécifiques :

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

Par exemple, la déclaration de stratégie suivante vous permet d'ouvrir un tunnel vers l'objet IoT nommé `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

L'action de stratégie `iot:OpenTunnel` prend en charge les clés de condition suivantes :

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `aws:RequestTag/`*`tag-key`*
- `aws:SecureTransport`
- `aws:TagKeys`

La déclaration de politique suivante vous permet d'ouvrir un tunnel vers l'objet si l'objet appartient à un groupe d'objets dont le nom commence par et `TestGroup` si le service de destination configuré sur le tunnel est SSH.

```
{
```

```
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ]
        }
    }
}
```

Vous pouvez également utiliser des balises de ressources pour contrôler l'autorisation d'ouvrir des tunnels. Par exemple, la déclaration de stratégie suivante permet d'ouvrir un tunnel si la clé de balise `Owner` est présente et que sa valeur est `Admin` et qu'aucune autre balise n'est spécifiée. Pour obtenir des informations sur comment utiliser les , consultez Marquer vos ressources AWS IoT.

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Owner": "Admin"
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "Owner"
        }
    }
}
```

IoT : RotateTunnelAccessToken

L'action de stratégie `iot:RotateTunnelAccessToken` accorde à un mandataire l'autorisation d'appeler RotateTunnelAccessToken.

Dans l'`Resource`élément de la déclaration de politique IAM :

- Spécifiez un ARN de tunnel entièrement qualifié :

  `arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

  Vous pouvez également utiliser l'ARN du tunnel générique :

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`
- Spécifiez un objet ARN pour gérer les `RotateTunnelAccessToken` autorisations pour des objets IoT spécifiques :

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

Par exemple, la déclaration de politique suivante vous permet de faire pivoter le jeton d'accès source d'un tunnel ou le jeton d'accès de destination d'un client pour l'objet IoT nommé `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

L'action de stratégie `iot:RotateTunnelAccessToken` prend en charge les clés de condition suivantes :

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `iot:ClientMode`
- `aws:SecureTransport`

La déclaration de politique générale suivante vous permet de faire pivoter le jeton d'accès de destination vers l'objet si l'objet appartient à un groupe d'objets dont le nom commence par `TestGroup`, le service de destination configuré sur le tunnel est SSH, et le client est en `DESTINATION` mode.

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ],
            "iot:ClientMode": "DESTINATION"
        }
    }
}
```

IoT : DescribeTunnel

L'action de stratégie `iot:DescribeTunnel` accorde à un mandataire l'autorisation d'appeler DescribeTunnel.

Dans l'`Resource`élément de la déclaration de politique IAM, spécifiez un ARN de tunnel entièrement qualifié :

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Vous pouvez également utiliser le caractère générique ARN :

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'action de stratégie `iot:DescribeTunnel` prend en charge les clés de condition suivantes :

- `aws:ResourceTag/tag-key`

- `aws:SecureTransport`

La déclaration de stratégie suivante vous permet d'appeler `DescribeTunnel` si le tunnel demandé est marqué avec la clé `Owner` ayant la valeur `Admin`.

```
{
    "Effect": "Allow",
    "Action": "iot:DescribeTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Owner": "Admin"
        }
    }
}
```

IoT : ListTunnels

L'action de stratégie `iot:ListTunnels` accorde à un mandataire l'autorisation d'appeler ListTunnels.

Dans l'`Resource`élément de la déclaration de politique IAM :

- Spécifiez l'ARN du tunnel générique :

  `arn:aws:iot:aws-region:aws-account-id:tunnel/*`
- Spécifiez un objet ARN pour gérer les `ListTunnels` autorisations sur les objets IoT sélectionnés :

  `arn:aws:iot:aws-region:aws-account-id:thing/thing-name`

L'`iot:ListTunnels`action politique soutient la clé de condition `aws:SecureTransport`.

La déclaration de stratégie suivante vous permet de répertorier les tunnels pour l'objet nommé `TestDevice`.

```
{
    "Effect": "Allow",
    "Action": "iot:ListTunnels",
```

```
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

IoT : ListTagsForResource

L'action de stratégie `iot:ListTagsForResource` accorde à un mandataire l'autorisation d'appeler `ListTagsForResource`.

Dans l'`Resource`élément de la déclaration de politique IAM, spécifiez un ARN de tunnel entièrement qualifié :

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Vous pouvez également utiliser l'ARN du tunnel générique :

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'action `iot:ListTagsForResource` politique soutient la clé de condition `aws:SecureTransport`.

IoT : CloseTunnel

L'action de stratégie `iot:CloseTunnel` accorde à un mandataire l'autorisation d'appeler CloseTunnel.

Dans l'`Resource`élément de la déclaration de politique IAM, spécifiez un ARN de tunnel entièrement qualifié :

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Vous pouvez également utiliser l'ARN du tunnel générique :

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'action de stratégie `iot:CloseTunnel` prend en charge les clés de condition suivantes :

- `iot:Delete`
- `aws:ResourceTag/tag-key`

- `aws:SecureTransport`

La déclaration de stratégie suivante vous permet d'appeler `CloseTunnel` si le paramètre `Delete` de la demande est `false` et si le tunnel demandé est balisé avec une clé `Owner` ayant la valeur `QATeam`.

```
{
    "Effect": "Allow",
    "Action": "iot:CloseTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "Bool": {
            "iot:Delete": "false"
        },
        "StringEquals": {
            "aws:ResourceTag/Owner": "QATeam"
        }
    }
}
```

IoT : TagResource

L'action de stratégie `iot:TagResource` accorde à un mandataire l'autorisation d'appeler `TagResource`.

Dans l'`Resource`élément de la déclaration de politique IAM, spécifiez un ARN de tunnel entièrement qualifié :

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

Vous pouvez également utiliser l'ARN du tunnel générique :

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

L'action `iot:TagResource` politique soutient la clé de condition `aws:SecureTransport`.

IoT : UntagResource

L'action de stratégie `iot:UntagResource` accorde à un mandataire l'autorisation d'appeler `UntagResource`.

Dans l'`Resource`élément de la déclaration de politique IAM, spécifiez un ARN de tunnel entièrement qualifié :

`arn:aws:iot:`*`aws-region`*`: `*`aws-account-id`*`:tunnel/`*`tunnel-id`*

Vous pouvez également utiliser l'ARN du tunnel générique :

`arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`

L'action `iot:UntagResource` politique soutient la clé de condition `aws:SecureTransport`.

Résolution des problèmes de connectivité liés au tunneling AWS IoT sécurisé en faisant pivoter les jetons d'accès client

Lorsque vous utilisez le tunneling AWS IoT sécurisé, vous pouvez rencontrer des problèmes de connectivité même si le tunnel est ouvert. Les sections suivantes présentent certains problèmes possibles et indiquent comment les résoudre en faisant pivoter les jetons d'accès client. Pour faire pivoter le jeton d'accès client (CAT), utilisez l'RotateTunnelAccessTokenAPI ou le rotate-tunnel-access-token AWS CLI. Selon que vous rencontrez une erreur lors de l'utilisation du client en mode source ou en mode destination, vous pouvez faire pivoter le CAT en mode source ou en mode destination, ou les deux.

> **Note**
>
> - Si vous ne savez pas si le CAT doit être pivoté sur la source ou sur la destination, vous pouvez le faire pivoter à la fois sur la source et sur la destination en réglant sur `ClientMode` ALL lorsque vous utilisez l'`RotateTunnelAccessToken`API.
> - La rotation du CAT ne prolonge pas la durée du tunnel. Par exemple, supposons que la durée du tunnel soit de 12 heures et que le tunnel soit déjà ouvert depuis 4 heures. Lorsque vous alternez les jetons d'accès, les nouveaux jetons générés ne peuvent être utilisés que pendant les 8 heures restantes.

Rubriques

- Erreur de jeton d'accès client non valide
- Erreur de non-concordance du jeton client

- Problèmes de connectivité de l'appareil à distance

Erreur de jeton d'accès client non valide

Lorsque vous utilisez le tunneling AWS IoT sécurisé, vous pouvez rencontrer une erreur de connexion lorsque vous utilisez le même jeton d'accès client (CAT) pour vous reconnecter au même tunnel. Dans ce cas, le proxy local ne peut pas se connecter au serveur proxy de tunneling sécurisé. Si vous utilisez un client en mode source, le message d'erreur suivant peut s'afficher :

```
Invalid access token: The access token was previously used and cannot be used again
```

L'erreur se produit parce que le jeton d'accès client (CAT) ne peut être utilisé qu'une seule fois par le proxy local, puis il devient invalide. Pour résoudre cette erreur, faites pivoter le jeton d'accès client dans le `SOURCE` mode afin de générer un nouveau CAT pour la source. Pour voir un exemple qui montre comment faire pivoter le CAT source, consultezExemple CAT de rotation de la source.

Erreur de non-concordance du jeton client

> **Note**
>
> Il n'est pas recommandé d'utiliser des jetons clients pour réutiliser le CAT. Nous vous recommandons plutôt d'utiliser l'`RotateTunnelAccessToken`API pour faire pivoter les jetons d'accès client afin de vous reconnecter au tunnel.

Si vous utilisez des jetons clients, vous pouvez réutiliser le CAT pour vous reconnecter au tunnel. Pour réutiliser le CAT, vous devez fournir le jeton client au CAT la première fois que vous vous connectez au tunneling sécurisé. Le tunneling sécurisé stocke le jeton client. Ainsi, pour les tentatives de connexion ultérieures utilisant le même jeton, le jeton client doit également être fourni. Pour plus d'informations sur l'utilisation des jetons client, consultez l'implémentation de référence du proxy local dans GitHub.

Lorsque vous utilisez des jetons client, si vous utilisez un client en mode source, le message d'erreur suivant peut s'afficher :

```
Invalid client token: The provided client token does not match the client token
    that was previously set.
```

L'erreur se produit parce que le jeton client fourni ne correspond pas au jeton client fourni avec le CAT lors de l'accès au tunnel. Pour résoudre cette erreur, faites pivoter le CAT dans le `SOURCE` mode afin de générer un nouveau CAT pour la source. Voici un exemple:

Exemple CAT de rotation de la source

L'exemple suivant montre comment exécuter l'`RotateTunnelAccessToken`API dans le `SOURCE` mode permettant de générer un nouveau CAT pour la source :

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --region <region> \
    --tunnel-id <tunnel-id> \
    --client-mode SOURCE
```

L'exécution de cette commande génère un nouveau jeton d'accès à la source et renvoie l'ARN de votre tunnel.

```
{
    "sourceAccessToken": "<source-access-token>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

Vous pouvez désormais utiliser le nouveau jeton source pour connecter le proxy local en mode source.

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=<source-access-token>
./localproxy -r <region> -s <port>
```

Voici un exemple de résultat de l'exécution du proxy local :

```
...

[info]    Starting proxy in source mode
...
[info]    Successfully established websocket connection with proxy server ...
[info]    Listening for new connection on port <port>
...
```

Problèmes de connectivité de l'appareil à distance

Lorsque vous utilisez le tunneling AWS IoT sécurisé, l'appareil peut se déconnecter de façon inattendue même si le tunnel est ouvert. Pour savoir si un appareil est toujours connecté au tunnel, vous pouvez utiliser l'DescribeTunnelAPI ou le AWS CLI describe-tunnel.

Un appareil peut être déconnecté pour plusieurs raisons. Pour résoudre le problème de connectivité, vous pouvez faire pivoter le CAT sur la destination si l'appareil a été déconnecté pour les raisons suivantes :

- Le CAT de destination est devenu invalide.
- Le jeton n'a pas été livré à l'appareil via le sujet MQTT réservé au tunneling sécurisé :

  `$aws/things/<thing-name>/tunnels/notify`

L'exemple suivant montre comment résoudre ce problème :

Exemple CAT de rotation de la destination

Envisagez un appareil distant,*<RemoteThing1>*. Pour ouvrir un tunnel à cet effet, vous pouvez utiliser la commande suivante :

```
aws iotsecuretunneling open-tunnel \
    --region <region> \
    --destination-config thingName=<RemoteThing1>,services=SSH
```

L'exécution de cette commande génère les détails du tunnel et le CAT pour votre source et votre destination.

```
{
    "sourceAccessToken": "<source-access-token>",
    "destinationAccessToken": "<destination-access-token>",
    "tunnelId": "<tunnel-id>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/tunnel-id"
}
```

Toutefois, lorsque vous utilisez l'DescribeTunnelAPI, le résultat indique que l'appareil a été déconnecté, comme illustré ci-dessous :

```
aws iotsecuretunneling describe-tunnel \
```

```
    --tunnel-id <tunnel-id> \
    --region <region>
```

L'exécution de cette commande indique que le périphérique n'est toujours pas connecté.

```
{
    "tunnel": {
        ...
        "destinationConnectionState": {
            "status": "DISCONNECTED"
        },
        ...
    }
}
```

Pour résoudre cette erreur, exécutez l'`RotateTunnelAccessToken`API avec le client en `DESTINATION` mode et les configurations pour la destination. L'exécution de cette commande révoque l'ancien jeton d'accès, génère un nouveau jeton et renvoie ce jeton à la rubrique MQTT :

`$aws/things/<thing-name>/tunnels/notify`

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

L'exécution de cette commande génère le nouveau jeton d'accès, comme indiqué ci-dessous. Le jeton est ensuite envoyé à l'appareil pour qu'il se connecte au tunnel, si l'agent du périphérique est correctement configuré.

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

Mise en service des appareils

AWS propose plusieurs méthodes différentes pour approvisionner un appareil et y installer des certificats clients uniques. Cette section décrit chaque méthode et explique comment sélectionner celle qui convient le mieux à votre solution IoT. Ces options sont décrites en détail dans le livre blanc intitulé Device Manufacturing and Provisioning with X.509 Certificates in AWS IoT Core.

Sélectionnez l'option qui convient le mieux à votre situation

- Vous pouvez installer des certificats sur les appareils IoT avant qu'ils ne soient délivrés

  Si vous pouvez installer en toute sécurité des certificats clients uniques sur vos appareils IoT avant qu'ils ne soient fournis à l'utilisateur final, vous devez utiliser le just-in-timeprovisionnement (JITP) ou l'just-in-timeenregistrement (JITR).

  À l'aide de JITP et JITR, l'autorité de certification (CA) utilisée pour signer le certificat de l'appareil est enregistrée AWS IoT et reconnue AWS IoT lors de la première connexion de l'appareil. L'appareil est approvisionné AWS IoT lors de sa première connexion à l'aide des détails de son modèle de provisionnement.

  Pour plus d'informations sur le single thing, le JITP, le JITR et la mise en service groupé d'appareils dotés de certificats uniques, veuillez consulter the section called "Mise en service d'appareils disposant de certificats d'appareils".
- Les utilisateurs finaux ou les installateurs peuvent utiliser une application pour installer des certificats sur leurs appareils IoT

  Si vous ne pouvez pas installer de manière sécurisée des certificats clients uniques sur votre appareil IoT avant qu'ils ne soient fournis à l'utilisateur final, mais que l'utilisateur final ou un installateur peut utiliser une application pour enregistrer les appareils et installer les certificats d'appareils uniques, vous devez utiliser le processus de mise en service par un utilisateur de confiance.

  L'utilisation d'un utilisateur de confiance, tel qu'un utilisateur final ou un installateur possédant un compte connu, peut simplifier le processus de fabrication des appareils. Au lieu d'un certificat client unique, les appareils disposent d'un certificat temporaire qui leur permet de se connecter AWS IoT pendant 5 minutes seulement. Au cours de cette fenêtre de 5 minutes, l'utilisateur de confiance obtient un certificat client unique avec une durée de vie plus longue et l'installe sur l'appareil. La durée de vie limitée du certificat de réclamation minimise le risque de compromission du certificat.

Pour de plus amples informations, veuillez consulter the section called “Allocation par utilisateur approuvé”.

- Les utilisateurs finaux NE PEUVENT PAS utiliser une application pour installer des certificats sur leurs appareils IoT

  Si aucune des options précédentes ne fonctionne dans votre solution IoT, la mise en service par processus de demande est une option. Dans le cadre de ce processus, vos appareils IoT disposent d’un certificat de réclamation qui est partagé par les autres appareils de la flotte. La première fois qu'un appareil se connecte à un certificat de réclamation, AWS IoT enregistre l'appareil à l'aide de son modèle de provisionnement et délivre à l'appareil son certificat client unique pour un accès ultérieur. AWS IoT

  Cette option permet le provisionnement automatique d'un appareil lorsqu'il se connecte à AWS IoT, mais peut présenter un risque plus important en cas de compromission d'un certificat de réclamation. Si un certificat de demande est compromis, vous pouvez le désactiver. La désactivation du certificat de demande empêche l’enregistrement futur de tous les appareils dotés de ce certificat de demande. Toutefois, la désactivation du certificat de demande ne bloque pas les appareils déjà mis en service.

  Pour de plus amples informations, veuillez consulter the section called “Allocation par revendication”.

Provisionnement d'appareils dans AWS IoT

Lorsque vous approvisionnez un appareil AWS IoT, vous devez créer des ressources pour que vos appareils AWS IoT puissent communiquer en toute sécurité. D'autres ressources peuvent être créées pour vous aider à gérer votre flotte d'appareils. Les ressources suivantes peuvent être créées au cours du processus de mise en service :

- Un objet IoT.

  Les objets IoT sont des entrées dans le registre des AWS IoT appareils. Chaque objet a un nom et un ensemble d'attributs uniques, et est associé à un appareil physique. Les objets peuvent être définis à l'aide d'un type d'objet ou regroupées en groupes d'objets. Pour de plus amples informations, veuillez consulter Gestion des appareils avec AWS IoT.

Bien qu'elle ne soit pas nécessaire, la création d'un objet vous permet de gérer votre flotte d'appareils plus efficacement en recherchant les appareils par type d'objet, groupe d'objets et attributs d'objet. Pour de plus amples informations, veuillez consulter Indexation de la flotte.

> **Note**
>
> Pour indexer les données d'état de connectivité de votre objet, approvisionnez votre objet et configurez-le de manière à ce que le nom de l'objet corresponde à l'ID client utilisé dans la demande Connect.

- Un certificat X.509.

  Les appareils utilisent des certificats X.509 pour effectuer une authentification mutuelle avec AWS IoT. Vous pouvez enregistrer un certificat existant ou faire AWS IoT générer et enregistrer un nouveau certificat pour vous. Vous associez un certificat à un appareil en l'attachant à l'objet qui représente l'appareil. Vous devez également copier le certificat et la clé privée associée sur l'appareil. Les appareils présentent le certificat lors de la connexion à AWS IoT. Pour de plus amples informations, veuillez consulter Authentification.

- Une stratégie IoT.

  Les stratégies IoT définissent les opérations qu'un appareil peut effectuer dans AWS IoT. Les stratégies IoT sont attachées aux certificats des appareils. Lorsqu'un appareil présente le certificat à AWS IoT, il reçoit les autorisations spécifiées dans la politique. Pour de plus amples informations, veuillez consulter Autorisation. Chaque appareil a besoin d'un certificat pour communiquer avec AWS IoT.

AWS IoT prend en charge le provisionnement automatique de la flotte à l'aide de modèles de provisionnement. Les modèles de provisionnement décrivent les ressources nécessaires AWS IoT pour approvisionner votre appareil. Les modèles contiennent des variables qui vous permettent d'utiliser un modèle pour mettre en service plusieurs appareils. Lorsque vous allouez un appareil, vous spécifiez des valeurs pour les variables spécifiques à l'appareil à l'aide d'un dictionnaire ou d'une carte. Pour mettre en service un autre appareil, spécifiez de nouvelles valeurs dans le dictionnaire.

Vous pouvez utiliser l'allocation automatisée, que vos appareils disposent de certificats uniques (et de leur clé privée associée) ou non.

Approvisionnement de flotte APIs

Il existe plusieurs catégories de produits APIs utilisés dans le provisionnement de flottes :

- Ces fonctions de plan de contrôle créent et gèrent les modèles de provisionnement de flotte et configurent les stratégies des utilisateurs approuvés.
  - CreateProvisioningTemplate
  - CreateProvisioningTemplateVersion
  - DeleteProvisioningTemplate
  - DeleteProvisioningTemplateVersion
  - DescribeProvisioningTemplate
  - DescribeProvisioningTemplateVersion
  - ListProvisioningTemplates
  - ListProvisioningTemplateVersions
  - UpdateProvisioningTemplate
- Les utilisateurs approuvés peuvent utiliser cette fonction de plan de contrôle pour générer une demande d'intégration temporaire. Cette demande temporaire est transmise à l'appareil lors de la configuration Wi-Fi ou d’une méthode similaire.
  - CreateProvisioningClaim
- API MQTT utilisée au cours du processus de provisionnement par des périphériques avec un certificat de demande de provisionnement incorporé dans un périphérique ou transmis par un utilisateur approuvé.
  - the section called “CreateCertificateFromCsr”
  - the section called “CreateKeysAndCertificate”
  - the section called “RegisterThing”

Mise en service d'appareils qui ne disposent pas de certificats d'appareils à l'aide de la mise en service de flotte

En utilisant le provisionnement du AWS IoT parc, AWS IoT vous pouvez générer et délivrer en toute sécurité des certificats d'appareils et des clés privées à vos appareils lorsqu'ils se connectent AWS IoT pour la première fois. AWS IoT fournit des certificats clients signés par l'autorité de certification Amazon Root (CA).

Il existe deux façons d'utiliser la mise en service de flotte :

- Allocation par revendication
- Allocation par utilisateur approuvé

Allocation par revendication

Les appareils peuvent être fabriqués avec un certificat de revendication de mise en service et une clé privée (qui sont des informations d'identification à usage spécial) intégrés. Si ces certificats sont enregistrés AWS IoT, le service peut les échanger contre des certificats d'appareil uniques que l'appareil peut utiliser pour des opérations régulières. Le processus comprend les étapes suivantes :

Avant de livrer l'appareil

1. Appelez `CreateProvisioningTemplate` pour créer un modèle de mise en service. Cette API renvoie un ARN de modèle. Pour de plus amples informations, veuillez consulter API MQTT de mise en service des appareils.

   Vous pouvez également créer un modèle de provisionnement de flotte dans la AWS IoT console.

   a. Dans le volet de navigation, choisissez le menu déroulant Connect many devices. Choisissez ensuite Connect many devices.
   b. Choisissez Créer un modèle de provisionnement.
   c. Choisissez le scénario de provisionnement le mieux adapté à vos processus d'installation. Ensuite, choisissez Suivant.
   d. Terminez le flux de travail du modèle.
2. Créez les certificats et les clés privées associées qui seront utilisés comme certificats de revendications de mise en service.
3. Enregistrez ces certificats AWS IoT et associez-leur une politique IoT qui restreint l'utilisation des certificats. L'exemple suivant de stratégie IoT limite l'utilisation du certificat associé à cette stratégie aux appareils mis en service.

   JSON

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

```
        {
            "Effect": "Allow",
            "Action": ["iot:Connect"],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Publish","iot:Receive"],
            "Resource": [
                "arn:aws:iot:aws-region:aws-account-id:topic/$aws/
certificates/create/*",
                "arn:aws:iot:aws-region:aws-account-id:topic/$aws/
provisioning-templates/templateName/provision/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:aws:iot:aws-region:aws-account-id:topicfilter/$aws/
certificates/create/*",
                "arn:aws:iot:aws-region:aws-account-id:topicfilter/$aws/
provisioning-templates/templateName/provision/*"
            ]
        }
    ]
}
```

4. Lorsque vous approvisionnez des appareils, autorisez le AWS IoT service à créer ou à mettre à jour des ressources IoT telles que des objets et des certificats dans votre compte. Pour ce faire, associez la politique `AWSIoTThingsRegistration` gérée à un rôle IAM (appelé rôle de provisionnement) qui fait confiance au principal du AWS IoT service.
5. Fabriquez l'appareil avec le certificat de revendication de mise en service intégré de manière sécurisée.

L'appareil est maintenant prêt à être livré là où il sera installé pour être utilisé.

> **⚠ Important**
>
> Les clés privées des revendications de mise en service doivent être sécurisées à tout moment, y compris sur l'appareil. Nous vous recommandons d'utiliser AWS IoT CloudWatch

des métriques et des journaux pour détecter les signes d'utilisation abusive. Si vous détectez une utilisation abusive, désactivez le certificat de revendication de mise en service afin qu'il ne puisse pas être utilisé pour la mise en service des appareils.

Pour initialiser l'appareil à utiliser

1. L'appareil utilise le AWS IoT Appareil SDKs, appareil mobile SDKs et client AWS IoT d'appareil pour se connecter et s'authentifier à AWS IoT l'aide du certificat de demande de provisionnement installé sur l'appareil.

   > **Note**
   >
   > Pour des raisons de sécurité, le s articles `certificateOwnershipToken` retournés par `CreateCertificateFromCsr` et `CreateKeysAndCertificate` expire au bout d'une heure. `RegisterThing` doit être appelé avant l'expiration `certificateOwnershipToken`. Si le certificat créé par `CreateCertificateFromCsr` ou `CreateKeysAndCertificate` n'a pas été activé et n'a pas été attaché à une politique ou à un objet au moment où le jeton expire, le certificat est supprimé. Si le jeton expire, l'appareil peut appeler `CreateCertificateFromCsr` ou `CreateKeysAndCertificate` à nouveau pour générer un nouveau certificat.

2. L'appareil obtient un certificat permanent et une clé privée en utilisant l'une de ces options. L'appareil utilisera le certificat et la clé pour toutes les futures authentifications avec AWS IoT.

   a. Appelez `CreateKeysAndCertificate`pour créer un nouveau certificat et une nouvelle clé privée à l'aide de l'autorité de AWS certification.

      Ou

   b. Appelez `CreateCertificateFromCsr` pour générer un certificat à partir d'une demande de signature de certificat qui conserve sa clé privée sécurisée.

3. À partir de l'appareil, appelez `RegisterThing` pour enregistrer ce dernier auprès d' AWS IoT et créer des ressources cloud.

   Le service de mise en service de flotte utilise un modèle de mise en service pour définir et créer des ressources en cloud telles que des objets IoT. Le modèle peut spécifier les attributs et les

groupes auxquels appartient l'objet. Les groupes d'objets doivent exister avant que le nouvel objet puisse y être ajouté.

4. Après avoir enregistré le certificat permanent sur l'appareil, celui-ci doit se déconnecter de la session qu'il a ouverte avec le certificat de revendication de mise en service et se reconnecter à l'aide du certificat permanent.

L'appareil est maintenant prêt à communiquer normalement avec AWS IoT.

Allocation par utilisateur approuvé

Dans de nombreux cas, un appareil se connecte AWS IoT pour la première fois lorsqu'un utilisateur de confiance, tel qu'un utilisateur final ou un technicien d'installation, utilise une application mobile pour configurer l'appareil dans son emplacement de déploiement.

> **Important**
>
> Vous devez gérer l'accès et l'autorisation de l'utilisateur approuvé pour effectuer cette procédure. Une façon de le faire consiste à fournir et à gérer un compte pour l'utilisateur approuvé qui l'authentifie et lui accorde l'accès aux fonctionnalités AWS IoT et aux opérations d'API requises pour effectuer cette procédure.

Avant de livrer l'appareil

1. Appelez CreateProvisioningTemplatepour créer un modèle de provisionnement et renvoyer son *templateArn* et*templateName*.
2. Créez un rôle IAM qui permettra à un utilisateur de confiance de lancer le processus de mise en service. Le modèle de mise en service permet uniquement à cet utilisateur de mettre en service un appareil. Par exemple :

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:provisioningtemplate/templateName"
    ]
```

```
}
```

3. Lorsque vous approvisionnez des appareils, autorisez le AWS IoT service à créer ou à mettre à jour des ressources IoT, telles que des objets et des certificats dans votre compte. Pour ce faire, vous devez associer la politique `AWSIoTThingsRegistration` gérée à un rôle IAM (appelé rôle de provisionnement) qui fait confiance au principal du AWS IoT service.
4. Fournissez les moyens d'identifier vos utilisateurs de confiance, par exemple en leur fournissant un compte capable de les authentifier et d'autoriser leurs interactions avec les opérations d' AWS API nécessaires à l'enregistrement de leurs appareils.

Pour initialiser l’appareil à utiliser

1. Un utilisateur de confiance se connecte à votre application mobile ou service web de mise en service.
2. L’application mobile ou l’application web utilise le rôle IAM et appelle `CreateProvisioningClaim` pour obtenir un certificat de revendication de mise en service temporaire auprès AWS IoT.

   > **Note**
   >
   > Pour des raisons de sécurité, le certificat de demande de mise en service temporaire que `CreateProvisioningClaim` renvoie expire au bout de cinq minutes. Les étapes suivantes doivent renvoyer un certificat valide avant l'expiration du certificat de revendication de mise en service temporaire. Les certificats de revendication de mise en service temporaire n'apparaissent pas dans la liste des certificats de votre compte.

3. L'application mobile ou l'application web fournit le certificat de revendication de mise en service à l'appareil ainsi que toutes les informations de configuration nécessaires, notamment les informations d'identification Wi-Fi.
4. L'appareil utilise le certificat de revendication de mise en service temporaire pour se connecter à AWS IoT à l'aide du AWS IoT Appareil SDKs, appareil mobile SDKs et client AWS IoT d'appareil.
5. L'appareil obtient un certificat permanent et une clé privée en utilisant l'une de ces options dans les cinq minutes suivant la connexion au AWS IoT certificat de demande de provisionnement temporaire. L'appareil utilisera le certificat et la clé renvoyés par ces options pour toute future authentification AWS IoT.

a. Appelez `CreateKeysAndCertificate`pour créer un nouveau certificat et une nouvelle clé privée à l'aide de l'autorité de AWS certification.

   Ou

b. Appelez `CreateCertificateFromCsr` pour générer un certificat à partir d'une demande de signature de certificat qui conserve sa clé privée sécurisée.

> **Note**
>
> N'oubliez pas `CreateKeysAndCertificate`ou `CreateCertificateFromCsr`devez renvoyer un certificat valide dans les cinq minutes suivant la connexion AWS IoT au certificat de demande de provisionnement temporaire.

6. L'appareil appelle `RegisterThing`pour enregistrer l'appareil auprès de celui-ci AWS IoT et créer des ressources cloud.

   La service de mise en service de flotte utilise un modèle de mise en service pour définir et créer des ressources en cloud telles que des objets IoT. Le modèle peut spécifier les attributs et les groupes auxquels appartient l'objet. Les groupes d'objets doivent exister avant que le nouvel objet puisse y être ajouté.

7. Après avoir enregistré le certificat permanent sur l'appareil, celui-ci doit se déconnecter de la session qu'il a ouverte avec le certificat de revendication de mise en service temporaire et se reconnecter à l'aide du certificat permanent.

L'appareil est maintenant prêt à communiquer normalement avec AWS IoT.

Utilisation des hooks de pré-provisionnement avec l'interface de ligne de commande AWS

La procédure suivante crée un modèle de mise en service avec des hooks de mise en service en amont. La fonction Lambda utilisée ici est un exemple qui peut être modifié.

Pour créer et appliquer un hook de mise en service en amont à un modèle de mise en service

1. Créez une fonction Lambda dont l'entrée et la sortie sont définies. Les fonctions Lambda sont hautement personnalisables. `allowProvisioning` et `parameterOverrides` sont

nécessaires pour créer des hooks de pré-mise en service. Pour plus d'informations sur la création de fonctions Lambda, consultez la section Utilisation AWS Lambda avec l'interface de ligne de AWS commande.

Voici un exemple de sortie de fonction Lambda :

```
{
  "allowProvisioning": True,
  "parameterOverrides": {
    "incomingKey0": "incomingValue0",
    "incomingKey1": "incomingValue1"
  }
}
```

2. AWS IoT utilise des politiques basées sur les ressources pour appeler Lambda. Vous devez donc AWS IoT autoriser l'appel de votre fonction Lambda.

> **Important**
>
> Assurez-vous d'inclure le `source-arn` ou `source-account` dans les clés de contexte de condition globale des politiques associées à votre action Lambda afin d'empêcher toute manipulation des autorisations. Pour de plus amples informations à ce sujet, veuillez consulter Prévention du cas de figure de l'adjoint désorienté entre services.

L'exemple suivant utilise add-permission pour donner à IoT l'autorisation d'appeler votre fonction Lambda.

```
aws lambda add-permission \
    --function-name myLambdaFunction \
    --statement-id iot-permission \
    --action lambda:InvokeFunction \
    --principal iot.amazonaws.com
```

3. Ajoutez un hook de pré-provisionnement à un modèle à l'aide de la commande create-provisioning-templateor update-provisioning-template.

L'exemple de CLI suivant utilise le create-provisioning-templatepour créer un modèle de provisioning doté de hooks de pré-provisionnement :

```
aws iot create-provisioning-template \
    --template-name myTemplate \
    --provisioning-role-arn arn:aws:iam:us-east-1:1234564789012:role/myRole \
    --template-body file://template.json \
    --pre-provisioning-hook file://hooks.json
```

La sortie de cette commande ressemble à ce qui suit :

```
{
    "templateArn": "arn:aws:iot:us-east-1:1234564789012:provisioningtemplate/
myTemplate",
    "defaultVersionId": 1,
    "templateName": myTemplate
}
```

Pour gagner du temps, vous pouvez également charger un paramètre à partir d'un fichier au lieu de le taper en tant que valeur de paramètre de ligne de commande. Pour plus d'informations, consultez Loading Chargement AWS CLI from a File paramètres à partir d'un fichier. Le code suivant illustre le paramètre `template` au format JSON étendu :

```
{
    "Parameters" : {
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
```

```
                "ThingName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingTypeName" : {"Fn::Join":["",["ThingTypePrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["widgets", "WA"],
                "BillingGroup": "BillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:504350838278:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
    }
}
```

Le code suivant illustre le paramètre `pre-provisioning-hook` au format JSON étendu :

```
{
     "targetArn" : "arn:aws:lambda:us-
east-1:765219403047:function:pre_provisioning_test",
     "payloadVersion" : "2020-04-01"
}
```

Mise en service d'appareils disposant de certificats d'appareils

AWS IoT propose trois méthodes pour approvisionner des appareils lorsqu'ils disposent déjà d'un certificat d'appareil (et d'une clé privée associée) :

- Mise en service d'un objet unique avec un modèle de mise en service. Cette option convient si vous devez uniquement mettre en service des appareils un par un.
- Just-in-time provisionnement (JITP) avec un modèle qui approvisionne un appareil lors de sa première connexion à. AWS IoT Cette option convient si vous devez enregistrer un grand nombre d'appareils, mais que vous n'avez pas d'informations sur ceux-ci que vous pouvez assembler dans une liste de mise en service en bloc.
- Enregistrement en bloc. Cette option vous permet de spécifier une liste de valeurs de modèle de mise en service d'objet unique qui sont stockées dans un fichier au sein d'un compartiment S3. Cette approche fonctionne bien si vous avez un grand nombre d'appareils connus dont vous pouvez assembler les caractéristiques souhaitées dans une liste.

Rubriques

- Mise en service d'un seul objet
- Just-in-time approvisionnement
- Enregistrement en bloc

Mise en service d'un seul objet

Pour provisionner un objet, utilisez l'RegisterThingAPI ou la commande `register-thing` CLI. La commande de l'interface de ligne de commande `register-thing` accepte les arguments suivants :

--template-body

Modèle de mise en service.

--parameters

Liste de paires nom-valeur pour les paramètres utilisés dans le modèle de mise en service, au format JSON (par exemple, `{"ThingName" : "MyProvisionedThing", "CSR" : "csr-text"}`).

Consultez Mise en service des modèles.

RegisterThingou `register-thing` renvoie le ARNs pour les ressources et le texte du certificat qu'il a créé :

```
{
    "certificatePem": "certificate-text",
    "resourceArns": {
    "PolicyLogicalName": "arn:aws:iot:us-
west-2:123456789012:policy/2A6577675B7CD1823E271C7AAD8184F44630FFD7",
    "certificate": "arn:aws:iot:us-west-2:123456789012:cert/
cd82bb924d4c6ccbb14986dcb4f40f30d892cc6b3ce7ad5008ed6542eea2b049",
    "thing": "arn:aws:iot:us-west-2:123456789012:thing/MyProvisionedThing"
    }
}
```

Si un paramètre est omis du dictionnaire, la valeur par défaut est utilisée. Si aucune valeur par défaut n'est spécifiée, le paramètre n'est pas remplacé par une valeur.

Just-in-time approvisionnement

Vous pouvez utiliser le just-in-time provisionnement (JITP) pour approvisionner vos appareils lors de leur première tentative de connexion. AWS IoT Pour mettre en service l'appareil, vous devez activer l'enregistrement automatique et associer un modèle de mise en service au certificat CA utilisé pour signer le certificat d'appareil. Les réussites et les erreurs de provisionnement sont enregistrées comme Métriques de mise en service d'appareils sur Amazon CloudWatch.

Rubriques

- Présentation du JITP
- Enregistrer CA à l'aide d'un modèle de mise en service

- Enregistrer CA à l'aide d'un nom de modèle de mise en service

Présentation du JITP

Lorsqu'un appareil tente de se connecter à AWS IoT l'aide d'un certificat signé par un certificat CA enregistré, AWS IoT charge le modèle à partir du certificat CA et l'utilise pour appeler RegisterThing. Le flux de travail de mise en service JITP enregistre d'abord un certificat avec une valeur de statut `PENDING_ACTIVATION`. Lorsque le flux de mise en service d'appareil est terminé, le statut du certificat est modifié en `ACTIVE`.

AWS IoT définit les paramètres suivants que vous pouvez déclarer et référencer dans les modèles de provisionnement :

- `AWS::IoT::Certificate::Country`
- `AWS::IoT::Certificate::Organization`
- `AWS::IoT::Certificate::OrganizationalUnit`
- `AWS::IoT::Certificate::DistinguishedNameQualifier`
- `AWS::IoT::Certificate::StateName`
- `AWS::IoT::Certificate::CommonName`
- `AWS::IoT::Certificate::SerialNumber`
- `AWS::IoT::Certificate::Id`

Les valeurs de ces paramètres de modèle de mise en service sont limitées à ce que la mise en service JITP peut extraire du champ d'objet du certificat de l'appareil qui est mis en service. Le certificat doit contenir des valeurs pour tous les paramètres du corps du modèle. Le paramètre `AWS::IoT::Certificate::Id` fait référence à un ID généré en interne, et non un ID qui est contenu dans le certificat. Vous pouvez obtenir la valeur de cet ID à l'aide de la `principal()` fonction intégrée à une AWS IoT règle.

> **Note**
>
> Vous pouvez approvisionner des appareils à l'aide de la fonction de AWS IoT Core just-in-time provisionnement (JITP) sans avoir à envoyer l'intégralité de la chaîne de confiance lors de la première connexion d'un appareil à. AWS IoT Core La présentation du certificat CA

est facultative, mais l'appareil doit envoyer l'extension SNI (Indication du nom du serveur) lorsqu'il se connecte à AWS IoT Core.

Exemple de corps de modèle

Le fichier JSON suivant est un exemple de corps de modèle d'un modèle JITP complet.

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
```

```
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect
\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

Cet exemple de modèle déclare des valeurs pour les paramètres de mise en service `AWS::IoT::Certificate::CommonName`, `AWS::IoT::Certificate::SerialNumber`, `AWS::IoT::Certificate::Country` et `AWS::IoT::Certificate::Id` qui sont extraits du certificat et utilisés dans la section `Resources`. Le flux de travail JITP utilise ensuite ce modèle pour effectuer les actions suivantes :

- Enregistrer un certificat et définir son état sur PENDING_ACTIVE.
- Créer une ressource d'objet.
- Créer une ressource de stratégie.
- Attacher la stratégie au certificat.
- Attacher le certificat à l'objet.
- Mettre à jour le statut du certificat en ACTIVE.

Le provisionnement de l'appareil échoue si le certificat ne possède pas toutes les propriétés mentionnées dans la `Parameters` section du`templateBody`. Par exemple, s'il `AWS::IoT::Certificate::Country` est inclus dans le modèle, mais que le certificat ne possède aucune propriété `Country`, la mise en service de l'appareil échoue.

Vous pouvez également l'utiliser CloudTrail pour résoudre les problèmes liés à votre modèle JITP. Pour plus d'informations sur les métriques enregistrées sur Amazon CloudWatch, consultezMétriques de mise en service d'appareils. Pour plus d'informations sur les modèles de mise en service, voir Modèles de mise en service.

> **Note**
>
> Au cours du processus de provisionnement, le just-in-time provisionnement (JITP) appelle d'autres opérations d'API du plan AWS IoT de contrôle. Ces appels peuvent dépasser les AWS IoT quotas de limitation définis pour votre compte et entraîner des appels limités. Veuillez contacter le AWS service clientèle pour augmenter vos quotas de limitation, si nécessaire.

Enregistrer CA à l'aide d'un modèle de mise en service

Pour enregistrer une autorité de certification à l'aide d'un modèle de mise en service complet, procédez comme suit :

1. Enregistrez votre modèle de mise en service et les informations ARN du rôle, comme dans l'exemple suivant, sous forme de fichier JSON :

```
{
     "templateBody" : "{\r\n    \"Parameters\" : {\r\n
 \"AWS::IoT::Certificate::CommonName\": {\r\n           \"Type\": \"String\"\r
\n        },\r\n        \"AWS::IoT::Certificate::SerialNumber\": {\r\n
  \"Type\": \"String\"\r\n        },\r\n        \"AWS::IoT::Certificate::Country
\": {\r\n            \"Type\": \"String\"\r\n        },\r\n
 \"AWS::IoT::Certificate::Id\": {\r\n            \"Type\": \"String\"\r
\n        }\r\n    },\r\n    \"Resources\": {\r\n        \"thing\": {\r
\n            \"Type\": \"AWS::IoT::Thing\",\r\n            \"Properties
\": {\r\n                \"ThingName\": {\r\n                    \"Ref\":
 \"AWS::IoT::Certificate::CommonName\"\r\n                },\r\n
    \"AttributePayload\": {\r\n                    \"version\": \"v1\",\r\n
                \"serialNumber\": {\r\n                        \"Ref\":
```

```
 \"AWS::IoT::Certificate::SerialNumber\"\r\n                    }\r\n
    },\r\n                \"ThingTypeName\": \"lightBulb-versionA\",\r\n
       \"ThingGroups\": [\r\n                    \"v1-lightbulbs\",\r\n
          {\r\n                        \"Ref\": \"AWS::IoT::Certificate::Country
\"\r\n                    }\r\n                ]\r\n            },\r\n
  \"OverrideSettings\": {\r\n                \"AttributePayload\": \"MERGE\",\r\n
               \"ThingTypeName\": \"REPLACE\",\r\n                \"ThingGroups
\": \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"certificate\": {\r
\n            \"Type\": \"AWS::IoT::Certificate\",\r\n            \"Properties
\": {\r\n                \"CertificateId\": {\r\n                    \"Ref\":
 \"AWS::IoT::Certificate::Id\"\r\n                },\r\n                \"Status\":
 \"ACTIVE\"\r\n            },\r\n            \"OverrideSettings\": {\r\n
     \"Status\": \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"policy
\": {\r\n            \"Type\": \"AWS::IoT::Policy\",\r\n            \"Properties
\": {\r\n                \"PolicyDocument\": \"{ \\\"Version\\\": \\\"2012-10-17\
\\", \\\"Statement\\\": [{ \\\"Effect\\\": \\\"Allow\\\", \\\"Action\\\":[\\
\"iot:Publish\\\"], \\\"Resource\\\": [\\\"arn:aws:iot:us-east-1:123456789012:topic
\/foo\/bar\\\"] }] }\"\r\n            }\r\n        }\r\n    }\r\n}",
     "roleArn" : "arn:aws:iam::123456789012:role/JITPRole"
}
```

Dans cet exemple, la valeur du champ `templateBody` doit être un objet JSON spécifié comme une chaîne échappée et ne peut utiliser que les valeurs de la liste précédente. Vous pouvez utiliser différents outils pour créer l'objet JSON requis, par exemple `json.dumps` (Python) ou `JSON.stringify` (Node). La valeur du champ `roleARN` doit être l'ARN d'un rôle qui est le `AWSIoTThingsRegistration` attaché à celui-ci. De plus, votre modèle peut utiliser un `PolicyName` au lieu du `PolicyDocument` en ligne dans l'exemple.

2. Enregistrez un certificat CA à l'aide de l'opération Register CACertificate API ou de la commande `register-ca-certificate`CLI. Vous allez spécifier le répertoire du modèle de mise en service et les informations ARN du rôle que vous avez enregistrées à l'étape précédente :

   L'exemple suivant montre comment enregistrer un certificat CA en mode `DEFAULT` à l'aide du AWS CLI :

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
   verification-cert file://your-verification-cert
                  --set-as-active --allow-auto-registration --registration-config
    file://your-template
   ```

L'exemple suivant montre comment enregistrer un certificat CA en mode SNI_ONLY à l'aide du AWS CLI :

```
aws iot register-ca-certificate --ca-certificate file://your-ca-cert --certificate-
mode SNI_ONLY
                --set-as-active --allow-auto-registration --registration-config
 file://your-template
```

Pour en savoir plus, veuillez consulter Enregistrement des certificats CA.

3. (Facultatif) Mettez à jour les paramètres d'un certificat CA à l'aide de l'opération Update CACertificate API ou de la commande update-ca-certificateCLI.

   L'exemple suivant montre comment mettre à jour un certificat d'autorité de certification à l'aide AWS CLI :

```
aws iot update-ca-certificate --certificate-id caCertificateId
               --new-auto-registration-status ENABLE --registration-config
 file://your-template
```

Enregistrer CA à l'aide d'un nom de modèle de mise en service

Pour enregistrer CA à l'aide d'un nom de modèle de mise en service, procédez comme suit :

1. Enregistrez le corps de votre modèle de mise en service en tant que fichier JSON. Vous pouvez trouver un exemple de corps de modèle dans un exemple de corps de modèle.
2. Pour créer un modèle de provisionnement, utilisez l'CreateProvisioningTemplateAPI ou la commande create-provisioning-templateCLI :

```
aws iot create-provisioning-template --template-name your-template-name \
        --template-body file://your-template-body.json --type JITP \
        --provisioning-role-arn arn:aws:iam::123456789012:role/test
```

> **Note**
>
> Pour le just-in-time provisionnement (JITP), vous devez spécifier le type de modèle à utiliser `JITP` lors de la création du modèle de provisionnement. Pour plus d'informations sur le type de modèle, consultez CreateProvisioningTemplatela référence de l'AWS API.

3. Pour enregistrer l'autorité de certification avec le nom du modèle, utilisez l'CACertificateAPI Register ou la commande `register-ca-certificate`CLI :

```
aws iot register-ca-certificate --ca-certificate file://your-ca-cert --
verification-cert file://your-verification-cert \
        --set-as-active --allow-auto-registration --registration-config
 templateName=your-template-name
```

Enregistrement en bloc

Vous pouvez utiliser la commande `start-thing-registration-task` pour enregistrer les objets en bloc. Cette commande prend un modèle de mise en service, un nom de compartiment S3, un nom de clé et un ARN de rôle qui autorise l'accès au fichier dans le compartiment S3. Le fichier du compartiment S3 contient les valeurs utilisées pour remplacer les paramètres dans le modèle. Le fichier doit être au format JSON délimité par une nouvelle ligne. Chaque ligne contient toutes les valeurs des paramètres pour l'enregistrement d'un seul appareil. Par exemple :

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

Les opérations API suivantes liées à l'enregistrement en masse peuvent être utiles :

- ListThingRegistrationTasks: Répertorie les tâches actuelles de provisionnement d'objets en masse.
- DescribeThingRegistrationTask: fournit des informations sur une tâche spécifique d'enregistrement d'objets groupés.
- StopThingRegistrationTask: arrête une tâche d'enregistrement d'objets groupés.
- ListThingRegistrationTaskReports: Utilisé pour vérifier les résultats et les échecs d'une tâche d'enregistrement groupée d'objets.

> **Note**
>
> - Vous ne pouvez exécuter qu'une seule tâche d'enregistrement en bloc à la fois (par compte).
> - Les opérations d'enregistrement groupé font appel à d'autres opérations AWS IoT de l'API du plan de contrôle. Ces appels peuvent dépasser les AWS IoT quotas de limitation définis dans votre compte et provoquer des erreurs de limitation. Contactez AWS le Support client pour augmenter vos AWS IoT quotas de régulation, si nécessaire.

Mise en service des modèles

Un modèle de provisionnement est un document JSON qui utilise des paramètres pour décrire les ressources avec AWS IoT lesquelles votre appareil doit interagir. Un modèle de mise en service contient deux sections : `Parameters` et `Resources`. Il existe deux types de modèles de provisionnement dans AWS IoT. L'un est utilisé pour le just-in-time provisionnement (JITP) et l'enregistrement en masse, et le second est utilisé pour le provisionnement de la flotte.

Rubriques

- Section Parameters
- Section Resources
- Exemple de modèle pour l’enregistrement en bloc
- Exemple de modèle pour le just-in-time provisionnement (JITP)
- Mise en service de flotte

Section Parameters

La section `Parameters` déclare les paramètres utilisés dans la section `Resources`. Chaque paramètre déclare un nom, un type et une valeur facultative par défaut. La valeur par défaut est utilisée lorsque le dictionnaire transmis avec le modèle ne contient pas de valeur pour le paramètre. La section `Parameters` d'un modèle de document ressemble à ce qui suit :

```
{
    "Parameters" : {
        "ThingName" : {
```

```
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    }
}
```

Cet extrait de corps de modèle déclare quatre paramètres : `ThingName`, `SerialNumber`, `Location` et `CSR`. Tous ces paramètres sont de type `String`. Le paramètre `Location` déclare la valeur par défaut `"WA"`.

Section Resources

La `Resources` section du corps du modèle indique les ressources nécessaires à la communication de votre appareil AWS IoT : un objet, un certificat et une ou plusieurs politiques IoT. Chaque ressource spécifie un nom logique, un type et un ensemble de propriétés.

Un nom logique vous permet de faire référence à une ressource à un autre endroit dans le modèle.

Le type spécifie le type de ressource que vous déclarez. Les types valides sont :

- `AWS::IoT::Thing`
- `AWS::IoT::Certificate`
- `AWS::IoT::Policy`

Les propriétés que vous spécifiez dépendent du type de ressource que vous déclarez.

Ressources d'objet

Les ressources d'objet sont déclarées à l'aide des propriétés suivantes :

- `ThingName`: String.

- `AttributePayload`: Facultatif. Liste de paires nom-valeur.
- `ThingTypeName`: Facultatif. Chaîne d'un type d'objet associé pour l'objet.
- `ThingGroups`: Facultatif. Liste des groupes auxquels l'objet appartient.
- `BillingGroup`: Facultatif. Chaîne pour le nom d’un groupe de facturation associé.
- `PackageVersions`: Facultatif. Chaîne pour un package associé et les noms de version.

Ressources de certificat

Les certificats peuvent être spécifiés de l'une des façons suivantes :

- Demande de signature du certificat (CSR).
- ID d'un certificat d'appareil existant. (Seul le certificat IDs peut être utilisé avec un modèle de provisionnement de flotte.)
- Certificat d'appareil créé avec un certificat CA enregistré auprès d' AWS IoT. Si vous avez plusieurs certificats CA enregistrés avec le même champ d'objet, vous devez également transmettre le certificat CA utilisé pour signer le certificat de l'appareil.

> **Note**
>
> Lorsque vous déclarez un certificat dans un modèle, vous devez uniquement utiliser ces méthodes. Par exemple, si vous utilisez une CSR, vous ne pouvez pas spécifier d'ID de certificat ou de certificat d'appareil. Pour de plus amples informations, veuillez consulter Certificats client X.509.

Pour de plus amples informations, veuillez consulter Présentation des certificats X.509.

Les ressources de certificat sont déclarées à l'aide des propriétés suivantes :

- `CertificateSigningRequest`: String.
- `CertificateId`: String.
- `CertificatePem`: String.
- `CACertificatePem`: String.
- `Status`: Facultatif. Chaîne qui peut être `ACTIVE` ou `INACTIVE`. ACTIVE est l'option sélectionnée par défaut.

- `ThingPrincipalType`: Facultatif. Chaîne qui indique le type de relation entre l'objet et le principal (le certificat).
  - `EXCLUSIVE_THING`: Établit une relation exclusive. Le principal ne peut être rattaché qu'à cette chose spécifique et à aucune autre.
  - `NON_EXCLUSIVE_THING`: Attache le principal spécifié aux objets. Vous pouvez associer plusieurs objets au principal. Il s'agit de la valeur par défaut si elle n'est pas spécifiée.

> **Note**
>
> Vous pouvez également approvisionner des appareils sans certificat d'appareil. Pour de plus amples informations, veuillez consulter Mise en service d'appareils qui ne disposent pas de certificats d'appareils à l'aide de la mise en service de flotte.

Exemples :

- Certificat spécifié avec une CSR :

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateSigningRequest": {"Ref" : "CSR"},
            "Status" : "ACTIVE"
        }
    }
}
```

- Certificat spécifié avec un ID de certificat existant :

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateId": {"Ref" : "CertificateId"}
        }
    }
}
```

- Certificat spécifié avec un fichier .pem de certificat et un fichier .pem de certificat de CA existants :

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CACertificatePem": {"Ref" : "CACertificatePem"},
            "CertificatePem": {"Ref" : "CertificatePem"}
        }
    }
}
```

- Attachez exclusivement une chose à un principal :

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "ThingPrincipalType" : "EXCLUSIVE_THING"
        }
    }
}
```

Ressources de politique

Les ressources de stratégie sont déclarées à l'aide de l'une des propriétés suivantes :

- `PolicyName`: Facultatif. String. Un hachage du document de stratégie est utilisé par défaut. Les `PolicyName` peuvent uniquement faire référence à des politiques AWS IoT , mais pas à des politiques IAM. Si vous utilisez une stratégie AWS IoT existante, pour la propriété `PolicyName`, saisissez le nom de la stratégie. N'incluez pas la propriété `PolicyDocument`.
- `PolicyDocument`: Facultatif. Un objet JSON spécifié comme une chaîne placée dans une séquence d'échappement. Si `PolicyDocument` n'est pas fourni, la stratégie doit déjà être créée.

> **Note**
>
> Si une section `Policy` est présente, `PolicyName` ou `PolicyDocument`, mais pas les deux, doit être spécifié.

Paramètres de remplacement

Si un modèle spécifie une ressource qui existe déjà, la section `OverrideSettings` vous permet de spécifier l'action à effectuer :

`DO_NOTHING`

Conserver la ressource telle qu'elle est.

`REPLACE`

Remplacer la ressource par celle qui est spécifiée dans le modèle.

`FAIL`

Entraîner l'échec de la demande avec `ResourceConflictsException`.

`MERGE`

Valide uniquement pour les propriétés `ThingGroups` et `AttributePayload` d'un `thing`. Fusionnez les attributs existants ou les appartenances aux groupes de l'objet avec ceux spécifiés dans le modèle.

Lorsque vous déclarez une ressource d'objet, vous pouvez spécifier `OverrideSettings` pour les propriétés suivantes :

- `ATTRIBUTE_PAYLOAD`
- `THING_TYPE_NAME`
- `THING_GROUPS`

Lorsque vous déclarez une ressource de certificat, vous pouvez spécifier `OverrideSettings` pour la propriété `Status`.

`OverrideSettings` n'est pas disponible pour les ressources de stratégie.

Exemple de ressource

L'extrait de modèle suivant déclare un objet, un certificat et une stratégie :

```
{
```

```
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\", \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

L'objet est déclaré avec :

- Le nom logique `"thing"`.
- Le type `AWS::IoT::Thing`.
- Un ensemble de propriétés d'objet.

  Les propriétés d'objet comprennent le nom de l'objet, un ensemble d'attributs, un nom de type d'objet facultatif et une liste facultative de groupes d'objets auxquels l'objet appartient.

Les paramètres sont référencés par `{"Ref":"parameter-name"}`. Lorsque le modèle est évalué, les paramètres sont remplacés par la valeur du paramètre à partir du dictionnaire transmis avec le modèle.

Le certificat est déclaré avec :

- Le nom logique `"certificate"`.
- Le type `AWS::IoT::Certificate`.
- Un ensemble de propriétés.

  Les propriétés incluent la CSR pour le certificat et la définition de l'état sur `ACTIVE`. Le texte de la CSR est transmis en tant que paramètre dans le dictionnaire transmis avec le modèle.

La stratégie est déclarée avec :

- Le nom logique `"policy"`.
- Le type `AWS::IoT::Policy`.
- Le nom d'une stratégie existante ou un document de stratégie.

Exemple de modèle pour l’enregistrement en bloc

Le fichier JSON suivant est un exemple de modèle de mise en service complet qui spécifie le certificat avec une CSR :

(La valeur du champ `PolicyDocument` doit être un objet JSON spécifié comme une chaîne placée dans une séquence d'échappement.)

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
```

```
        "CSR" : {
            "Type" : "String"
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\", \"Statement
\": [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\":
 [\"arn:aws:iot:us-east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

Exemple de modèle pour le just-in-time provisionnement (JITP)

Le fichier JSON suivant est un exemple de modèle de mise en service complet qui spécifie un certificat existant avec un ID de certificat :

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
```

```
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
```

```
            "Status":"ACTIVE",
            "ThingPrincipalType" : "EXCLUSIVE_THING"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect
\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

> **Important**
>
> Vous devez l'utiliser `CertificateId` dans un modèle utilisé pour la mise en service JIT.

Pour plus d'informations sur le type de modèle de provisionnement, consultez la référence `CreateProvisioningTemplate`de l' AWS API.

Pour plus d'informations sur l'utilisation de ce modèle pour le just-in-time provisionnement, voir : Just-in-time provisioning.

Mise en service de flotte

Les modèles de provisionnement de flotte sont utilisés AWS IoT pour configurer la configuration du cloud et des appareils. Ces modèles utilisent les mêmes paramètres et ressources que le JITP et les modèles d'enregistrement en bloc. Pour de plus amples informations, veuillez consulter Mise en service des modèles. Les modèles de mise en service de flotte peuvent contenir une section `Mapping` et une section `DeviceConfiguration`. Vous pouvez utiliser des fonctions intrinsèques à l'intérieur d'un modèle de mise en service de flotte pour générer une configuration spécifique à l'appareil. Les modèles de provisionnement de flotte sont nommés ressources et sont identifiés par ARNs (par exemple,`arn:aws:iot:us-west-2:1234568788:provisioningtemplate/`*`templateName`*).

Mappages

La section `Mappings` facultative associe à une clé à un ensemble de valeurs correspondantes portant un nom. Par exemple, si vous souhaitez définir des valeurs en fonction d'une AWS région, vous pouvez créer un mappage qui utilise le Région AWS nom comme clé et qui contient les valeurs que vous souhaitez spécifier pour chaque région spécifique. Pour récupérer les valeurs d'un mappage, utilisez la fonction intrinsèque `Fn::FindInMap`.

Vous ne pouvez pas inclure des paramètres, des pseudo-paramètres ou des fonctions d'appel intrinsèques dans la section `Mappings`.

Configuration de l'appareil

La section Configuration des appareils contient des données arbitraires que vous souhaitez envoyer à vos appareils lors de la mise en service. Par exemple :

```
{
    "DeviceConfiguration": {
        "Foo":"Bar"
    }
}
```

Si vous envoyez des messages à vos appareils à l'aide du format de charge utile JSON (JavaScript Object Notation), AWS IoT Core formatez ces données au format JSON. Si vous utilisez le format de charge utile CBOR (Représentation binaire concise des objets), AWS IoT Core formate ces données au format CBOR. La section `DeviceConfiguration` ne prend pas en charge les objets JSON imbriqués.

Fonctions intrinsèques

Les fonctions intrinsèques sont utilisées dans n'importe quelle section du modèle de mise en service, à l'exception de la section `Mappings`.

`Fn::Join`

> Ajoute un ensemble de valeurs dans une seule valeur, séparées par le délimiteur spécifié. Si un délimiteur est une chaîne vide, l’ensemble de valeurs est concaténé avec aucun délimiteur.

> **Important**
>
> `Fn::Join` n'est pas pris en charge pour the section called “Ressources de politique”.

`Fn::Select`

Renvoie un seul objet à partir d'une liste d'objets en fonction de son index.

> **Important**
>
> `Fn::Select` ne recherche pas les valeurs `null` ou ne vérifie pas si l'index sort des limites du tableau. Les deux conditions entraînent une erreur de provisionnement. Assurez-vous donc que vous avez choisi une valeur d'index valide et que la liste contient des valeurs non nulles.

`Fn::FindInMap`

Renvoie la valeur correspondant aux clés dans un mappage à deux niveaux déclaré dans la section `Mappings`.

`Fn::Split`

Divise une chaîne en liste de valeurs de chaîne afin que vous puissiez sélectionner un élément dans la liste de chaînes. Vous spécifiez un délimiteur qui détermine où la chaîne est fractionnée (par exemple, une virgule). Après avoir divisé une chaîne, utilisez `Fn::Select` pour sélectionner un élément.

Par exemple, si une chaîne de sous-réseau séparée par des virgules IDs est importée dans votre modèle de pile, vous pouvez diviser la chaîne à chaque virgule. Dans la liste des sous-réseaux IDs, utilisez cette option `Fn::Select` pour spécifier un ID de sous-réseau pour une ressource.

`Fn::Sub`

Remplace les variables contenues dans une chaîne d'entrée par des valeurs que vous spécifiez. Vous pouvez utiliser cette fonction pour construire des commandes ou des sorties qui incluent des valeurs qui ne sont pas disponibles tant que vous n'avez pas créé ou mis à jour une pile.

Exemple de modèle pour la mise en service d'une flotte

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber": {
            "Type": "String"
        },
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Ref" : "ThingName"},
                "ThingTypeName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["v1-lightbulbs", "WA"],
                "BillingGroup": "LightBulbBillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
```

```
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active",
                "ThingPrincipalType" : "EXCLUSIVE_THING"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
        }
}
```

> **Note**
>
> Un modèle de mise en service existant peut être mis à jour afin d'ajouter un hook de mise en service en amont.

Hooks de mise en service en amont

AWS recommande d'utiliser des fonctions de pré-approvisionnement lors de la création de modèles de provisionnement afin de mieux contrôler quels appareils et combien d'appareils sont intégrés à votre compte. Les hooks de mise en service en amont sont des fonctions Lambda qui valident les paramètres transmis par l'appareil avant d'autoriser la mise en service de l'appareil. Cette fonction

Lambda doit exister dans votre compte avant la mise en service d'un appareil, car elle est appelée chaque fois qu'un appareil envoie une demande via the section called "RegisterThing".

> **Important**
>
> Assurez-vous d'inclure le `source-arn` ou `source-account` dans les clés de contexte de condition globale des politiques associées à votre action Lambda afin d'empêcher toute manipulation des autorisations. Pour de plus amples informations à ce sujet, veuillez consulter Prévention du cas de figure de l'adjoint désorienté entre services.

Pour que les appareils soient provisionnés, votre fonction Lambda doit accepter l'objet d'entrée et renvoyer l'objet de sortie décrit dans cette section. La mise en service ne se déroule que si la fonction Lambda renvoie un objet avec la valeur `"allowProvisioning": True`.

Entrée du hook de pré-provisionnement

AWS IoT envoie cet objet à la fonction Lambda lorsqu'un appareil s'enregistre auprès de. AWS IoT

```
{
    "claimCertificateId" : "string",
    "certificateId" : "string",
    "certificatePem" : "string",
    "templateArn" : "arn:aws:iot:us-east-1:1234567890:provisioningtemplate/MyTemplate",
    "clientId" : "221a6d10-9c7f-42f1-9153-e52e6fc869c1",
    "parameters" : {
        "string" : "string",
        ...
    }
}
```

L'objet `parameters` transmis à la fonction Lambda contient les propriétés de l'argument `parameters` passé dans la charge utile de la demande the section called "RegisterThing".

Valeur de retour du hook de pré-provisionnement

La fonction Lambda doit renvoyer une réponse indiquant si elle a autorisé la demande de mise en service et les valeurs de toutes les propriétés à remplacer.

Voici un exemple de réponse réussie de la fonction de pré-provisionnement.

```
{
    "allowProvisioning": true,
    "parameterOverrides" : {
        "Key": "newCustomValue",
        ...
    }
}
```

Les valeurs `"parameterOverrides"` seront ajoutées au paramètre `"parameters"` dans la charge utile de la demande the section called “RegisterThing”.

> **Note**
>
> - Si la fonction Lambda échoue, la demande de provisionnement échoue `ACCESS_DENIED` et une erreur est enregistrée dans Logs. CloudWatch
> - Si la fonction Lambda ne renvoie pas `"allowProvisioning": "true"` dans la réponse, la demande de mise en service échoue avec `ACCESS_DENIED`.
> - La fonction Lambda doit procéder à l'exécution et renvoyer dans un délai de 5 secondes, sinon la demande de mise en service échoue.

Exemple Lambda de hook de mise en service

Python

Exemple de hook de mise en service en amont Lambda en Python.

```
import json

def pre_provisioning_hook(event, context):
    print(event)

    return {
        'allowProvisioning': True,
        'parameterOverrides': {
            'DeviceLocation': 'Seattle'
        }
    }
```

Java

Exemple de hook de mise en service en amont Lambda en Java.

Classe de gestionnaire :

```
package example;

import java.util.Map;
import java.util.HashMap;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class PreProvisioningHook implements
 RequestHandler<PreProvisioningHookRequest, PreProvisioningHookResponse> {

    public PreProvisioningHookResponse handleRequest(PreProvisioningHookRequest
 object, Context context) {
        Map<String, String> parameterOverrides = new HashMap<String, String>();
        parameterOverrides.put("DeviceLocation", "Seattle");

        PreProvisioningHookResponse response = PreProvisioningHookResponse.builder()
                .allowProvisioning(true)
                .parameterOverrides(parameterOverrides)
                .build();

        return response;
    }

}
```

Classe de demande :

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@Builder
```

```
@AllArgsConstructor
@NoArgsConstructor
public class PreProvisioningHookRequest {
    private String claimCertificateId;
    private String certificateId;
    private String certificatePem;
    private String templateArn;
    private String clientId;
    private Map<String, String> parameters;
}
```

La classe de réponse :

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class PreProvisioningHookResponse {
    private boolean allowProvisioning;
    private Map<String, String> parameterOverrides;
}
```

JavaScript

Exemple d'un hook de pré-provisionnement Lambda dans. JavaScript

```
exports.handler = function(event, context, callback) {
    console.log(JSON.stringify(event, null, 2));
    var reply = {
        allowProvisioning: true,
        parameterOverrides: {
            DeviceLocation: 'Seattle'
        }
     };
```

```
    callback(null, reply);
}
```

Signature de certificats autogérée à l'aide d' AWS IoT Core un fournisseur de certificats

Vous pouvez créer un fournisseur de AWS IoT Core certificats pour signer les demandes de signature de certificats (CSRs) lors du provisionnement de AWS IoT flotte. Un fournisseur de certificats fait référence à une fonction Lambda et à l'API `CreateCertificateFromCsr` MQTT pour le provisionnement de flottes. La fonction Lambda accepte un CSR et renvoie un certificat client signé.

Lorsque vous n'avez pas de fournisseur de certificats Compte AWS, l'API CreateCertificateFromCsr MQTT est appelée lors du provisionnement de la flotte pour générer le certificat à partir d'un CSR. Après avoir créé un fournisseur de certificats, le comportement de l'API CreateCertificateFromCsr MQTT changera et tous les appels à cette API MQTT appelleront le fournisseur de certificats pour émettre le certificat.

Avec le fournisseur de AWS IoT Core certificats, vous pouvez mettre en œuvre des solutions qui utilisent des autorités de certification privées (CAs) telles que AWS Private CAd'autres autorités publiques fiables CAs ou votre propre infrastructure à clé publique (PKI) pour signer le CSR. En outre, vous pouvez utiliser le fournisseur de certificats pour personnaliser les champs de votre certificat client tels que les périodes de validité, les algorithmes de signature, les émetteurs et les extensions.

> **Important**
>
> Vous ne pouvez créer qu'un seul fournisseur de certificats par Compte AWS. Le changement de comportement de signature s'applique à l'ensemble du parc qui appelle l'API CreateCertificateFromCsr MQTT jusqu'à ce que vous supprimiez le fournisseur de certificats de votre Compte AWS.

Dans cette rubrique :

- Comment fonctionne la signature de certificats autogérés dans le cadre de l'approvisionnement de flottes
- Entrée de la fonction Lambda du fournisseur de certificats

- Valeur renvoyée par la fonction Lambda du fournisseur de certificats
- Exemple de fonction Lambda
- Signature de certificats autogérée pour le provisionnement de la flotte
- AWS CLI commandes pour le fournisseur de certificats

Comment fonctionne la signature de certificats autogérés dans le cadre de l'approvisionnement de flottes

Concepts clés

Les concepts suivants fournissent des détails qui peuvent vous aider à comprendre le fonctionnement de la signature de certificats autogérés dans le cadre du provisionnement de AWS IoT flottes. Pour plus d'informations, consultez la section Provisionnement d'appareils ne disposant pas de certificats d'appareils à l'aide du provisionnement de flotte.

AWS IoT approvisionnement de flotte

Le provisionnement de AWS IoT flotte (abréviation de Fleet Provisioning) permet de AWS IoT Core générer et de délivrer en toute sécurité des certificats d'appareils à vos appareils lorsqu'ils se connectent AWS IoT Core pour la première fois. Vous pouvez utiliser le provisionnement du parc pour connecter des appareils qui ne possèdent pas de certificat d'appareil à AWS IoT Core.

Demande de signature de certificat (CSR)

Au cours du processus de provisionnement de flotte, un appareil fait une demande AWS IoT Core via le MQTT de provisionnement de flotte. APIs Cette demande inclut une demande de signature de certificat (CSR), qui sera signée pour créer un certificat client.

AWS signature de certificats gérés dans le cadre de l'approvisionnement de la flotte

AWS géré est le paramètre par défaut pour la signature des certificats dans le cadre du provisionnement de flotte. Avec la signature de certificat AWS gérée, elle AWS IoT Core signera CSRs en utilisant sa propre signature CAs.

Signature de certificats autogérés dans le cadre du provisionnement de la flotte

L'autogestion est une autre option pour la signature de certificats dans le cadre de l'approvisionnement de flottes. Avec la signature de certificats autogérée, vous créez un fournisseur de AWS IoT Core certificats à signer CSRs. Vous pouvez utiliser la signature de certificat autogérée pour signer CSRs avec une autorité de certification générée par une autorité

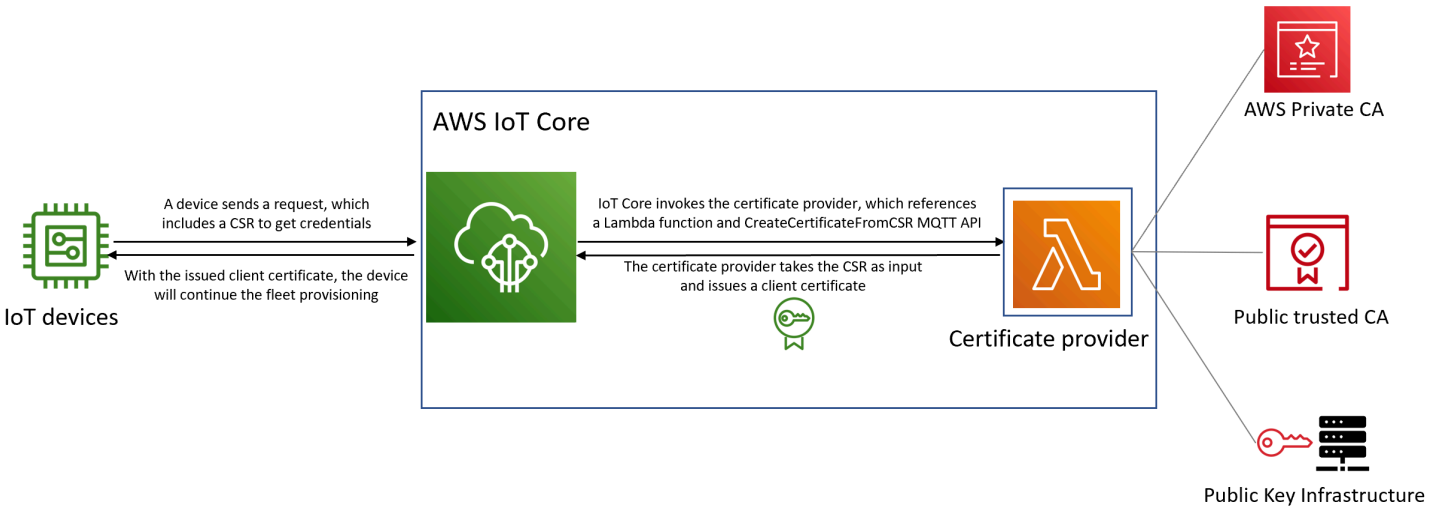
de certification AWS privée, une autre autorité de certification approuvée par le public ou votre propre infrastructure à clé publique (PKI).

AWS IoT Core fournisseur de certificats

AWS IoT Core un fournisseur de certificats (abréviation de fournisseur de certificats) est une ressource gérée par le client qui est utilisée pour la signature autogérée de certificats dans le cadre de l'approvisionnement de flottes.

Schéma

Le schéma suivant illustre de manière simplifiée le fonctionnement de la signature automatique des certificats dans le cadre du provisionnement de AWS IoT flottes.



- Lorsqu'un nouvel appareil IoT est fabriqué ou introduit dans le parc, il a besoin de certificats clients pour s'authentifier. AWS IoT Core
- Dans le cadre du processus de provisionnement de la flotte, l'appareil fait une demande de certificats client par le biais du MQTT de provisionnement de la flotte. AWS IoT Core APIs Cette demande inclut une demande de signature de certificat (CSR).
- AWS IoT Core invoque le fournisseur de certificats et transmet le CSR en entrée au fournisseur.
- Le fournisseur de certificats prend le CSR en entrée et émet un certificat client.

  Pour la signature de certificats AWS gérés, AWS IoT Core signe le CSR à l'aide de sa propre autorité de certification et émet un certificat client.
- Avec le certificat client émis, l'appareil poursuivra le provisionnement de la flotte et établira une connexion sécurisée avec AWS IoT Core.

Entrée de la fonction Lambda du fournisseur de certificats

AWS IoT Core envoie l'objet suivant à la fonction Lambda lorsqu'un appareil s'enregistre auprès de cette fonction. La valeur de `certificateSigningRequest` est le CSR au format PEM (Privacy-Enhanced Mail) fourni dans la demande. `CreateCertificateFromCsr` `principalId`Il s'agit de l'ID du principal utilisé pour se connecter AWS IoT Core lors de la `CreateCertificateFromCsr` demande. `clientId`est l'ID client défini pour la connexion MQTT.

```
{
 "certificateSigningRequest": "string",
 "principalId": "string",
 "clientId": "string"
}
```

Valeur renvoyée par la fonction Lambda du fournisseur de certificats

La fonction Lambda doit renvoyer une réponse contenant la `certificatePem` valeur. Voici un exemple de réponse réussie. AWS IoT Core utilisera la valeur de retour (`certificatePem`) pour créer le certificat.

```
{
 "certificatePem": "string"
}
```

Si l'enregistrement est réussi, il `CreateCertificateFromCsr` renverra la même chose `certificatePem` dans la `CreateCertificateFromCsr` réponse. Pour plus d'informations, consultez l'exemple de charge utile de réponse de CreateCertificateFromCsr.

Exemple de fonction Lambda

Avant de créer un fournisseur de certificats, vous devez créer une fonction Lambda pour signer un CSR. Voici un exemple de fonction Lambda en Python. Cette fonction appelle AWS Private CA à signer le CSR d'entrée, à l'aide d'une autorité de certification privée et de l'algorithme de `SHA256WITHRSA` signature. Le certificat client retourné sera valide pendant un an. Pour plus d'informations sur AWS Private CA et comment créer une autorité de certification privée, voir Qu'est-ce qu'une autorité de certification AWS privée ? et Création d'une autorité de certification privée.

```
import os
```

```
import time
import uuid
import boto3

def lambda_handler(event, context):
    ca_arn = os.environ['CA_ARN']
    csr = (event['certificateSigningRequest']).encode('utf-8')

    acmpca = boto3.client('acm-pca')
    cert_arn = acmpca.issue_certificate(
        CertificateAuthorityArn=ca_arn,
        Csr=csr,
        Validity={"Type": "DAYS", "Value": 365},
        SigningAlgorithm='SHA256WITHRSA',
        IdempotencyToken=str(uuid.uuid4())
    )['CertificateArn']

    # Wait for certificate to be issued
    time.sleep(1)
    cert_pem = acmpca.get_certificate(
        CertificateAuthorityArn=ca_arn,
        CertificateArn=cert_arn
    )['Certificate']

    return {
        'certificatePem': cert_pem
    }
```

Important

- Les certificats renvoyés par la fonction Lambda doivent avoir le même nom d'objet et la même clé publique que la demande de signature de certificat (CSR).
- La fonction Lambda doit s'exécuter dans 5 secondes.
- La fonction Lambda doit se trouver dans la même région que Compte AWS la ressource du fournisseur de certificats.
- Le principal du AWS IoT service doit disposer de l'autorisation d'appel pour la fonction Lambda. Pour éviter toute confusion liée aux adjoints, nous vous recommandons de définir `sourceArn` et `sourceAccount` pour les autorisations d'appel. Pour plus d'informations, consultez Prévention du problème de l'adjoint confus entre services.

L'exemple de politique basée sur les ressources suivant pour Lambda accorde AWS IoT l'autorisation d'invoquer la fonction Lambda :

JSON

```
{
 "Version": "2012-10-17",
 "Id": "InvokePermission",
 "Statement": [
  {
   "Sid": "LambdaAllowIotProvider",
   "Effect": "Allow",
   "Principal": {
    "Service": "iot.amazonaws.com"
   },
   "Action": "lambda:InvokeFunction",
   "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
   "Condition": {
    "StringEquals": {
     "AWS:SourceAccount": "123456789012"
    },
    "ArnLike": {
     "AWS:SourceArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
    }
   }
  }
 ]
}
```

Signature de certificats autogérée pour le provisionnement de la flotte

Vous pouvez choisir la signature de certificat autogérée pour le provisionnement de la flotte à l'aide AWS CLI de ou. AWS Management Console

AWS CLI

Pour choisir la signature de certificats autogérée, vous devez créer un fournisseur de AWS IoT Core certificats pour vous connecter CSRs au provisionnement du parc. AWS IoT Core invoque le fournisseur de certificats, qui prend un CSR en entrée et renvoie un certificat client. Pour créer un

fournisseur de certificats, utilisez l'opération `CreateCertificateProvider` API ou la commande `create-certificate-provider` CLI.

> **Note**
>
> Une fois que vous avez créé un fournisseur de certificats, le comportement de l'`CreateCertificateFromCsr`API pour le provisionnement de la flotte changera, de sorte que tous les appels vers `CreateCertificateFromCsr` invoqueront le fournisseur de certificats pour créer les certificats. La modification de ce comportement peut prendre quelques minutes après la création d'un fournisseur de certificats.

```
aws iot create-certificate-provider \
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-1 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

Voici un exemple de sortie pour cette commande :

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Pour plus d'informations, consultez la référence *`CreateCertificateProvider`* de AWS IoTl'API.

AWS Management Console

Pour choisir la signature autogérée des certificats à l'aide de AWS Management Console, procédez comme suit :

1. Accédez à la console AWS IoT.
2. Dans le menu de navigation de gauche, sous Sécurité, choisissez Signature du certificat.
3. Sur la page de signature du certificat, sous Détails de signature du certificat, choisissez Modifier la méthode de signature du certificat.
4. Sur la page Modifier la méthode de signature du certificat, sous Méthode de signature du certificat, sélectionnez Autogéré.

5. Dans la section Paramètres autogérés, entrez le nom du fournisseur de certificats, puis créez ou choisissez une fonction Lambda.
6. Choisissez Mettre à jour la signature du certificat.

AWS CLI commandes pour le fournisseur de certificats

Création d'un fournisseur de certificats

Pour créer un fournisseur de certificats, utilisez l'opération `CreateCertificateProvider` API ou la commande `create-certificate-provider` CLI.

> **Note**
>
> Une fois que vous avez créé un fournisseur de certificats, le comportement de l'`CreateCertificateFromCsr`API pour le provisionnement de la flotte changera, de sorte que tous les appels vers `CreateCertificateFromCsr` invoqueront le fournisseur de certificats pour créer les certificats. La modification de ce comportement peut prendre quelques minutes après la création d'un fournisseur de certificats.

```
aws iot create-certificate-provider \
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-
function-1 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

Voici un exemple de sortie pour cette commande :

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-
certificate-provider"
}
```

Pour plus d'informations, consultez la référence *`CreateCertificateProvider`* de AWS IoTl'API.

Mettre à jour le fournisseur de certificats

Pour mettre à jour un fournisseur de certificats, utilisez l'opération `UpdateCertificateProvider` API ou la commande `update-certificate-provider` CLI.

```
aws iot update-certificate-provider \
                --certificateProviderName my-certificate-provider \
                --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-function-2 \
                --accountDefaultForOperations CreateCertificateFromCsr
```

Voici un exemple de sortie pour cette commande :

```
{
 "certificateProviderName": "my-certificate-provider",
 "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

Pour plus d'informations, consultez la référence *UpdateCertificateProvider* de AWS IoTl'API.

Décrire le fournisseur de certificats

Pour décrire un fournisseur de certificats, utilisez l'opération `DescribeCertificateProvider` API ou la commande `describe-certificate-provider` CLI.

```
aws iot describe-certificate-provider --certificateProviderName my-certificate-provider
```

Voici un exemple de sortie pour cette commande :

```
{
 "certificateProviderName": "my-certificate-provider",
 "lambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
 "accountDefaultForOperations": [
  "CreateCertificateFromCsr"
 ],
 "creationDate": "2022-11-03T00:15",
 "lastModifiedDate": "2022-11-18T00:15"
}
```

Pour plus d'informations, consultez la référence *DescribeCertificateProvider* de AWS IoTl'API.

Supprimer le fournisseur de certificats

Pour supprimer un fournisseur de certificats, utilisez l'opération `DeleteCertificateProvider` API ou la commande `delete-certificate-provider` CLI. Si vous supprimez la ressource du fournisseur de certificats, le comportement de `CreateCertificateFromCsr` AWS IoT reprendra et des certificats signés par AWS IoT un CSR seront créés.

```
aws iot delete-certificate-provider --certificateProviderName my-certificate-provider
```

Cette commande ne produit aucune sortie.

Pour plus d'informations, consultez la référence *DeleteCertificateProvider* de AWS IoTl'API.

Liste des fournisseurs de certificats

Pour répertorier les fournisseurs de certificats qu' Compte AWS il contient, utilisez l'opération `ListCertificateProviders` API ou la commande `list-certificate-providers` CLI.

```
aws iot list-certificate-providers
```

Voici un exemple de sortie pour cette commande :

```
{
 "certificateProviders": [
  {
   "certificateProviderName": "my-certificate-provider",
   "certificateProviderArn": "arn:aws:iot:us-
east-1:123456789012:certificateprovider:my-certificate-provider"
  }
 ]
}
```

Pour plus d'informations, consultez la référence *ListCertificateProvider*de AWS IoTl'API.

Création de politiques et de rôles IAM pour un utilisateur installant un appareil

> **Note**
>
> Ces procédures ne sont destinées à être utilisées que sur instruction de la AWS IoT console. Pour accéder à cette page depuis la console, ouvrez créer un nouveau modèle de mise en service.

Pourquoi cela ne peut-il pas être fait dans la console AWS IoT ?

Pour une expérience plus sécurisée, les actions IAM sont effectuées dans la console IAM. Les procédures décrites dans cette section vous indiquent les étapes de création des rôles et politiques IAM nécessaires à l'utilisation du modèle de mise en service.

Création d'une politique IAM pour l'utilisateur qui installera un appareil

Cette procédure décrit comment créer une politique IAM qui autorise un utilisateur à installer un appareil à l'aide d'un modèle de mise en service.

Au cours de cette procédure, vous allez passer de la console IAM à la AWS IoT console. Nous vous recommandons d'ouvrir les deux consoles en même temps pendant que vous effectuez cette procédure.

Pour créer une politique IAM pour l'utilisateur qui installera un appareil

1. Ouvrez la page Centre de politiques dans la console IAM.
2. Choisissez Create Policy (Créer une politique).
3. Sur la page Créer une politique, choisissez l'onglet JSON.
4. Accédez à la page de la AWS IoT console où vous avez choisi Configurer la politique et le rôle des utilisateurs.
5. Dans la politique de mise en service des échantillons, choisissez Copier.
6. Revenez à la console IAM.
7. Dans l'éditeur JSON, collez la politique que vous avez copiée depuis la AWS IoT console. Cette politique est spécifique au modèle que vous créez dans la AWS IoT console.

8. Pour continuer, choisissez Suivant : Balises.
9. Sur la page Ajouter des balises (facultatif), choisissez Ajouter une balise pour chaque balise que vous souhaitez ajouter à cette politique. Vous pouvez ignorer cette étape si vous n'avez aucune balise à ajouter.
10. Choisissez Suivant : Vérification pour continuer.
11. Sur la page Examiner une stratégie, procédez comme suit :

    a. Dans Nom*, entrez un nom pour la politique qui vous aidera à vous souvenir de l'objectif de la politique.

       Notez le nom que vous donnez à cette politique car vous l'utiliserez dans la procédure suivante.

    b. Vous pouvez choisir de saisir une description facultative pour la politique que vous créez.

    c. Consultez le reste de cette politique et ses balises.
12. Choisissez Créer une stratégie afin de finaliser la création de la stratégie.

Après avoir créé votre nouvelle stratégie, continuez vers the section called "Création d'un rôle IAM pour l'utilisateur qui installera un appareil" pour créer l'entrée de rôle de l'utilisateur à laquelle vous joindrez cette stratégie.

Création d'un rôle IAM pour l'utilisateur qui installera un appareil

Ces étapes décrivent comment créer un rôle IAM qui authentifie l'utilisateur qui installera un appareil à l'aide d'un modèle de mise en service.

Pour créer une politique IAM pour l'utilisateur qui installera un appareil

1. Ouvrez le centre de Rôles dans la console IAM.
2. Sélectionnez Créer un rôle.
3. Dans Sélectionner une entité d'approbation, choisissez le type d'entité d'approbation à laquelle vous souhaitez donner accès au modèle que vous créez.
4. Choisissez ou entrez l'identification de l'entité d'approbation à laquelle vous souhaitez accorder l'accès, puis choisissez Suivant.
5. Dans la page Ajouter des autorisations dans Stratégies d'autorisations, dans la zone de recherche, saisissez le nom de la stratégie que vous avez créée lors de la procédure précédente.

6. Pour la liste de stratégies, choisissez la stratégie que vous avez créée lors de la procédure précédente, puis choisissez Suivant.
7. Dans la section Nommer, vérifier et créer, procédez comme suit :

   a. Pour Nom du rôle, saisissez un nom de rôle vous permettant de vous rappeler du but de ce rôle.
   b. Dans Description, vous pouvez choisir de saisir une description facultative du rôle. L'utilisation de cette étape n'est pas obligatoire pour continuer.
   c. Passez en revue les valeurs des étapes 1 et 2.
   d. Pour Ajouter des balises (facultatif), vous pouvez choisir d'ajouter des balises à ce rôle. L'utilisation de cette étape n'est pas obligatoire pour continuer.
   e. Vérifiez que les informations de cette page sont complètes et correctes, puis choisissez Créer un rôle.

Après avoir créé le nouveau rôle, revenez à la AWS IoT console pour continuer à créer le modèle.

Mettre à jour une politique existante pour autoriser un nouveau modèle

Les étapes suivantes décrivent comment ajouter un nouveau modèle à une politique IAM qui autorise un utilisateur à installer un appareil à l'aide d'un modèle de provisioning.

Pour ajouter un nouveau modèle à une politique IAM existante

1. Ouvrez la page Centre de politiques dans la console IAM.
2. Dans le champ de recherche, saisissez le nom de la politique à mettre à jour.
3. Dans la liste située sous le champ de recherche, recherchez la politique que vous souhaitez mettre à jour et choisissez le nom de la politique.
4. Pour le résumé des politiques, choisissez l'onglet JSON, si ce panneau n'est pas déjà visible.
5. Si vous souhaitez modifier la politique, choisissez Modifier la politique.
6. Dans l'éditeur, choisissez l'onglet JSON, si ce panneau n'est pas déjà visible.
7. Dans le document de politique, recherchez la déclaration de politique qui contient l'action `iot:CreateProvisioningClaim`.

   Si le document de politique ne contient aucune déclaration de politique avec l'action `iot:CreateProvisioningClaim`, copiez l'extrait d'instruction suivant et collez-le en tant qu'entrée supplémentaire dans le tableau `Statement` du document de politique.

> **Note**
>
> Cet extrait doit être placé avant le dernier caractère `]` du tableau `Statement`. Vous devrez peut-être ajouter une virgule avant ou après cet extrait pour corriger toute erreur de syntaxe.

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "--PUT YOUR NEW TEMPLATE ARN HERE--"
    ]
}
```

8. Accédez à la page de la AWS IoT console où vous avez choisi Modifier les autorisations du rôle utilisateur.
9. Recherchez l'ARN de ressource du modèle et choisissez Copier.
10. Revenez à la console IAM.
11. Collez le nom de ressource Amazon (ARN) copié en haut de la liste des modèles ARNs du `Statement` tableau afin qu'il s'agisse de la première entrée.

    S'il s'agit du seul ARN du tableau, supprimez la virgule à la fin de la valeur que vous venez de coller.
12. Passez en revue la déclaration de politique mise à jour et corrigez les erreurs signalées par l'éditeur.
13. Pour enregistrer le document de politique mis à jour, choisissez Passer en revue la politique.
14. Choisissez Passer en revue une stratégie, puis Enregistrer les modifications.
15. Retournez à la AWS IoT console.

API MQTT de mise en service des appareils

Le service Fleet Provisioning prend en charge les opérations d'API MQTT suivantes :

- the section called “CreateCertificateFromCsr”
- the section called “CreateKeysAndCertificate”
- the section called “RegisterThing”

Cette API prend en charge les tampons de réponse au format CBOR (Concise Binary JavaScript Object Representation) et au format JSON (Object Notation), selon le *payload-format* sujet. Pour plus de clarté, les exemples de réponse et de demande présentés dans cette section sont présentés au format JSON.

payload-format	Type de données du format de réponse
CBOR	CBOR (Concise Binary Object Representation, représentation concise d'objets binaires)
json	JavaScript Notation d'objets (JSON)

> **Important**
>
> Avant de publier une rubrique de message de demande, abonnez-vous aux rubriques de réponse pour recevoir la réponse. Les messages utilisés par cette API utilisent le protocole de publication et d'abonnement de MQTT pour fournir une interaction de demande et réponse.
> Si vous ne vous abonnez pas aux sujets de réponse avant de publier une demande, il est possible que vous ne receviez pas les résultats de cette demande.

CreateCertificateFromCsr

Crée un certificat à partir d'une demande de signature de certificat (CSR). AWS IoT fournit des certificats clients signés par l'autorité de certification Amazon Root (CA). Le nouveau certificat a un statut `PENDING_ACTIVATION`. Lorsque vous appelez `RegisterThing` pour provisionner un objet avec ce certificat, l'état du certificat devient `ACTIVE` ou `INACTIVE` comme décrit dans le modèle.

Pour plus d’informations sur la création d’un certificat client à l’aide de votre certificat d’autorité de certification et d’une demande de signature de certificat, reportez-vous à Création d’un certificat client à l'aide de votre certificat d’autorité de certification.

> **Note**
>
> Pour des raisons de sécurité, le `certificateOwnershipToken` renvoyé par `CreateCertificateFromCsr` expire au bout d'une heure. `RegisterThing` doit être appelé avant l'expiration de `certificateOwnershipToken`. Si le certificat créé par `CreateCertificateFromCsr` n'a pas été activé et attaché à une politique ou à un objet au moment où le jeton expire, le certificat est supprimé. Si le jeton expire, l'appareil peut appeler `CreateCertificateFromCsr` pour générer un nouveau certificat.

Demande d'CreateCertificateFromCsr

Publiez un message avec la rubrique `$aws/certificates/create-from-csr/`*`payload-format`*.

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

CreateCertificateFromCsrcharge utile de la demande

```
{
    "certificateSigningRequest": "string"
}
```

`certificateSigningRequest`

La CSR, au format PEM.

Réponse de CreateCertificateFromCsr

S'abonner à `$aws/certificates/create-from-csr/`*`payload-format`*`/accepted`

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

CreateCertificateFromCsrcharge utile de réponse

```
{
    "certificateOwnershipToken": "string",
    "certificateId": "string",
    "certificatePem": "string"
}
```

`certificateOwnershipToken`

Le jeton pour prouver la propriété du certificat lors de la mise en service.

`certificateId`

ID du certificat. Les opérations de gestion de certificats prennent uniquement en compte un ID de certificat.

`certificatePem`

Données du certificat, au format PEM.

CreateCertificateFromCsr erreur

Pour recevoir des réponses d'erreur, abonnez-vous à `$aws/certificates/create-from-csr/`*`payload-format`*`/rejected`.

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

CreateCertificateFromCsrcharge utile d'erreur

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

Le code de statut.

`errorCode`

Code de l'erreur.

`errorMessage`

Message d'erreur.

CreateKeysAndCertificate

Crée de nouvelles clés et un certificat. AWS IoT fournit des certificats clients signés par l'autorité de certification Amazon Root (CA). Le nouveau certificat a un statut `PENDING_ACTIVATION`. Lorsque vous appelez `RegisterThing` pour provisionner un objet avec ce certificat, l'état du certificat devient `ACTIVE` ou `INACTIVE` comme décrit dans le modèle.

> **Note**
>
> Pour des raisons de sécurité, le `certificateOwnershipToken` renvoyé par `CreateKeysAndCertificate` expire au bout d'une heure. `RegisterThing` doit être appelé avant l'expiration de `certificateOwnershipToken`. Si le certificat créé par `CreateKeysAndCertificate` n'a pas été activé et attaché à une politique ou à un objet au moment où le jeton expire, le certificat est supprimé. Si le jeton expire, l'appareil peut appeler `CreateKeysAndCertificate` pour générer un nouveau certificat.

Demande d'CreateKeysAndCertificate

Publiez un message sur `$aws/certificates/create/`*`payload-format`* avec une charge utile de message vide.

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

Réponse de CreateKeysAndCertificate

S'abonner à `$aws/certificates/create/`*`payload-format`*`/accepted`

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

Réponse de CreateKeysAndCertificate

```
{
    "certificateId": "string",
    "certificatePem": "string",
    "privateKey": "string",
    "certificateOwnershipToken": "string"
}
```

`certificateId`

ID du certificat.

`certificatePem`

Données du certificat, au format PEM.

`privateKey`

Clé privée.

`certificateOwnershipToken`

Le jeton pour prouver la propriété du certificat lors de la mise en service.

CreateKeysAndCertificate erreur

Pour recevoir des réponses d'erreur, abonnez-vous à `$aws/certificates/create/`*`payload-format`*`/rejected`.

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

CreateKeysAndCertificatecharge utile d'erreur

```
{
```

```
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

Le code de statut.

`errorCode`

Code de l'erreur.

`errorMessage`

Message d'erreur.

RegisterThing

Alloue un objet à l'aide d'un modèle prédéfini.

RegisterThing demande

Publier un message sur `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

`templateName`

Nom du modèle de mise en service.

RegisterThing charge utile de la demande

```
{
    "certificateOwnershipToken": "string",
    "parameters": {
        "string": "string",
        ...
    }
```

```
}
```

`certificateOwnershipToken`

Le jeton qui prouve la propriété du certificat. AWS IoT génère le jeton lorsque vous créez un certificat via MQTT.

`parameters`

Facultatif. Paires clé-valeur du périphérique utilisées par les hooks de pré-provisionnement pour évaluer la demande d'enregistrement.

RegisterThing réponse

S'abonner à `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/accepted`

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

`templateName`

Nom du modèle de mise en service.

RegisterThing charge utile de réponse

```
{
    "deviceConfiguration": {
        "string": "string",
        ...
    },
    "thingName": "string"
}
```

`deviceConfiguration`

Configuration de l'appareil définie dans le modèle.

`thingName`

Nom de l'objet IoT créé lors de la mise en service.

RegisterThing réponse d'erreur

Pour recevoir des réponses d'erreur, abonnez-vous à `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/rejected`.

`payload-format`

Format de charge utile du message en tant que `cbor` ou `json`.

`templateName`

Nom du modèle de mise en service.

RegisterThing charge utile de réponse aux erreurs

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

Le code de statut.

`errorCode`

Code de l'erreur.

`errorMessage`

Message d'erreur.

Indexation de la flotte

Vous pouvez utiliser l'indexation de la flotte pour indexer, rechercher et agréger les données de vos appareils à partir des sources suivantes : AWS IoT registre, AWS IoT Device Shadow, AWS IoT connectivité, catalogue de packages logiciels de gestion des AWS IoT appareils et AWS IoT Device Defenderviolations. Vous pouvez interroger un groupe d'appareils et agréger des statistiques sur les enregistrements des appareils en fonction de différentes combinaisons d'attributs de l'appareil, notamment l'état, la connectivité et les violations. Grâce à l'indexation de la flotte, vous pouvez organiser, étudier et dépanner votre flotte d'appareils.

L'indexation de flotte offre les fonctionnalités suivantes.

Gestion des mises à jour des index

Vous pouvez configurer un index de flotte pour indexer les mises à jour relatives à vos groupes d'objets, à vos registres d'objets, à la surveillance des appareils, à la connectivité des appareils et aux violations des appareils. Lorsque vous activez l'indexation de la flotte, vous AWS IoT créez un index pour vos objets ou groupes d'objets. `AWS_Things` est l'index créé pour toutes vos objets. `AWS_ThingGroups` est l'index qui contient tous vos groupes d'objets. Une fois l'indexation de la flotte activée, vous pouvez exécuter des requêtes sur votre index. Par exemple, vous pouvez trouver tous les appareils portatifs dont l'autonomie de la batterie est supérieure à 70 %. AWS IoT met à jour l'index en permanence avec vos dernières données. Pour de plus amples informations, veuillez consulter Managing fleet indexing (Gestion de l’indexation de la flotte.

Interrogation de l'état de connectivité d'un appareil spécifique

Cette API fournit un accès à faible latence et à haut débit aux informations de connectivité spécifiques aux appareils les plus récentes. Pour plus d'informations, consultez la section État de la connectivité de l'appareil.

Recherche parmi les sources de données

Vous pouvez créer une chaîne de requête basée sur un langage de requête et l'utiliser pour effectuer des recherches dans différentes sources de données. Vous devez également configurer les sources de données dans le paramètre d'indexation de la flotte afin que la configuration d'indexation contienne les sources de données à partir desquelles vous souhaitez effectuer une recherche.
La chaîne de requête décrit les éléments que vous souhaitez rechercher. Vous pouvez créer des

requêtes à l'aide de champs AWS gérés, de champs personnalisés et de tout attribut issu de vos sources de données indexées. Pour plus d'informations sur les sources de données qui prennent en charge l'indexation de flotte, consultez la section Gestion de l'indexation des objets.

Interrogation des données agrégées

Vous pouvez rechercher sur vos appareils des données globales et renvoyer des statistiques, un centile, une cardinalité ou une liste d'éléments avec des requêtes de recherche sur des champs particuliers. Vous pouvez exécuter des agrégations sur des champs AWS gérés ou sur tout attribut que vous configurez en tant que champs personnalisés dans les paramètres d'indexation du parc. Pour plus d'informations sur les requêtes d'agrégation, consultez la section Interrogation des données agrégées.

Surveillance des données agrégées et création d'alarmes à l'aide des indicateurs de flotte

Vous pouvez utiliser les métriques de flotte pour envoyer CloudWatch automatiquement des données agrégées, analyser les tendances et créer des alarmes afin de surveiller l'état global de votre flotte en fonction de seuils prédéfinis. Pour plus d'informations sur les métriques de la flotte, veuillez consulter Métriques de la flotte.

Gestion de l'indexation de la flotte

L'indexation de flotte gère deux types d'index pour vous : l'indexation des objets et l'indexation des groupes d'objets.

Indexation d'objets

L'index créé pour l'ensemble de vos objets est `AWS_Things`. L'indexation d'objet prend en charge les sources de données suivantes : données de AWS IoT registre, données AWS IoT Device Shadow, données de AWS IoT connectivité et données de AWS IoT Device Defenderviolations. En ajoutant ces sources de données à la configuration d'indexation de votre flotte, vous pouvez rechercher des objets, demander des données agrégées et créer des groupes d'objets dynamiques et des métriques de flotte en fonction de vos requêtes de recherche.

Registre :AWS IoT fournit un registre qui vous aide à gérer les choses. Vous pouvez ajouter les données du registre à la configuration d'indexation de votre flotte pour rechercher des appareils en

fonction des noms d'objets, des descriptions et d'autres attributs du registre. Pour plus d'informations sur le registre, consultez la section Comment gérer les objets avec le registre.

Shadow -Le service AWS IoT Device Shadow fournit des shadows qui vous aident à stocker les données d'état de votre appareil. L'indexation des objets prend en charge à la fois les shadows anonymes classiques et nommées. Pour indexer les shadows nommées, activez vos paramètres shadows nommées et spécifiez les noms de vos shadows dans la configuration d'indexation des objets. Par défaut, vous pouvez ajouter jusqu'à 10 noms d'ombres par zone Compte AWS. Pour savoir comment augmenter la limite du nombre de noms fictifs, consultez la section AWS IoT Device Management Quotas de la Référence AWS générale.

Pour ajouter des shadows nommées à des fins d'indexation :

- Si vous utilisez la AWS IoT console, activez l'indexation des objets, choisissez Ajouter des shadows nommées et ajoutez les noms de vos shadows via la sélection des shadows nommées.
- Si vous utilisez le AWS Command Line Interface (AWS CLI), définissez-le `namedShadowIndexingMode` comme étant `ON` et spécifiez les noms des ombres dans `IndexingFilter`. Pour voir des exemples de commandes CLI, consultez la section Gérer l'indexation des objets.

> **Important**
>
> Le 20 juillet 2022 est la version en disponibilité générale (GA) de l'intégration de l'indexation de la flotte de gestion des AWS IoT appareils avec des ombres AWS IoT Core nommées et de la AWS IoT Device Defender détection des violations. Avec cette version GA, vous pouvez indexer des shadows nommées spécifiques en spécifiant les noms des shadows. Si vous avez ajouté vos shadows nommées pour l'indexation pendant la période de préversion publique de cette fonctionnalité, du 30 novembre 2021 au 19 juillet 2022, nous vous encourageons à reconfigurer les paramètres d'indexation de votre flotte et à choisir des noms shadows spécifiques pour réduire les coûts d'indexation et optimiser les performances.

Pour plus d'informations sur les shadows, consultez AWS IoT Service Device Shadows.

Connectivité - Les données de connectivité des appareils vous aident à identifier l'état de connexion de vos appareils. Ces données de connectivité sont dictées par les événements du cycle de vie. Lorsqu'un client se connecte ou se déconnecte, AWS IoT publie les événements du cycle de vie avec des messages dans des rubriques MQTT. Un message de connexion ou de déconnexion peut être

une liste d'éléments JSON fournissant des détails sur l'état de la connexion. Pour plus d'informations sur la connectivité des appareils, consultez Événements du cycle de vie.

Violations de Device Defender : les données relatives aux AWS IoT Device Defender violations permettent d'identifier les comportements anormaux des appareils par rapport aux comportements normaux que vous définissez dans un profil de sécurité. Un profil de sécurité contient un ensemble de comportements attendus de l'appareil. Chaque comportement utilise une métrique qui indique le comportement normal de vos appareils. Pour plus d'informations sur les violations de Device Defender, consultez la section AWS IoT Device Defender détecter.

Pour de plus amples informations, veuillez consulter la section Gestion de l'indexation des objets.

Indexation du groupe d'objets

`AWS_ThingGroups` est l'index qui contient tous les groupes de votre objet. Cet index vous permet de rechercher des groupes en fonction de leur nom, de la description, des attributs et de tous les noms de groupes parents.

Pour de plus amples informations, veuillez consulter la section Gestion de l'indexation du groupe d'objet.

Champs gérés

Les champs gérés contiennent des données associées aux objets, aux groupes d'objets, aux ombres des appareils, à la connectivité des appareils et aux violations de Device Defender. AWS IoT définit le type de données dans les champs gérés. Vous spécifiez les valeurs de chaque champ géré lorsque vous créez un AWS IoT objet. Par exemple, les noms d'objets, les groupes d'objets et les descriptions d'objets sont tous des champs gérés. L'indexation de flotte indexe les champs gérés en fonction du mode d'indexation que vous spécifiez. Les champs gérés ne peuvent pas être modifiés ni s'afficher dans `customFields`. Pour plus d'informations, consultez Champs personnalisés.

La liste suivante répertorie les champs gérés pour l'indexation des objets :

- Champs gérés pour le registre

```
"managedFields" : [
  {name:thingId, type:String},
  {name:thingName, type:String},
  {name:registry.version, type:Number},
  {name:registry.thingTypeName, type:String},
```

```
  {name:registry.thingGroupNames, type:String},
]
```

- Champs gérés pour les shadows classiques sans nom

```
"managedFields" : [
  {name:shadow.version, type:Number},
  {name:shadow.hasDelta, type:Boolean}
]
```

- Champs gérés pour les shadows nommées

```
"managedFields" : [
  {name:shadow.name.shadowName.version, type:Number},
  {name:shadow.name.shadowName.hasDelta, type:Boolean}
]
```

- Champs gérés pour la connectivité d'objets

```
"managedFields" : [
  {name:connectivity.timestamp, type:Number},
  {name:connectivity.version, type:Number},
  {name:connectivity.connected, type:Boolean},
  {name:connectivity.disconnectReason, type:String}
]
```

- Champs gérés pour Device Defender

```
"managedFields" : [
  {name:deviceDefender.violationCount, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.metricName, type:String},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationTime, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationValue, type:String},
  {name:deviceDefender.securityprofile.behaviorname.inViolation, type:Boolean}
]
```

- Champs gérés pour les groupes d'objets

```
"managedFields" : [
  {name:description, type:String},
  {name:parentGroupNames, type:String},
  {name:thingGroupId, type:String},
  {name:thingGroupName, type:String},
```

```
  {name:version, type:Number},
]
```

Le tableau suivant répertorie les champs gérés qui ne sont pas consultables.

Source de données	Champ géré impossible à rechercher
Registre	`registry.version`
Shadows anonymes	`shadow.version`
Shadows nommés	`shadow.name.*.version`
Device Defender	`deviceDefender.version`
Groupes d'objets	`version`

Champs personnalisés

Vous pouvez agréger les attributs des objets, les données de Device Shadow et les données relatives aux violations de Device Defender en créant des champs personnalisés pour les indexer. L'attribut `customFields` est une liste de paires de noms de champs et de types de données. Vous pouvez effectuer des requêtes d'agrégation en fonction du type de données. Le mode d'indexation que vous choisissez et qui affecte les champs peut être spécifié dans `customFields`. Par exemple, si vous spécifiez le mode d'indexation `REGISTRY`, vous ne pouvez pas spécifier un champ personnalisé à partir d'une shadow d'objet. Vous pouvez utiliser la commande update-indexing-configurationCLI pour créer ou mettre à jour les champs personnalisés (voir un exemple de commande dans la section Mise à jour des exemples de configuration d'indexation).

- Noms de champs personnalisés

Les noms de champs personnalisés pour les attributs d'objets et de groupes d'objets commencent par `attributes.`, suivis du nom de l'attribut. Si l'indexation anonyme est activée, les éléments peuvent avoir des noms de champs personnalisés commençant par `shadow.desired` ou `shadow.reported`, suivis du nom de la valeur des données shadows sans nom. Si l'indexation de la shadow nommée est activée, les éléments peuvent avoir des noms de champs personnalisés commençant par `shadow.name.*.desired.` ou `shadow.name.*.reported.`, suivis de la valeur

des données shadows nommées. Si l'indexation des violations de Device Defender est activée, les éléments peuvent avoir des noms de champs personnalisés commençant par `deviceDefender.`, suivi de la valeur des données relatives aux violations de Device Defender.

Le nom de l'attribut ou de la valeur de données qui suit le préfixe ne peut contenir que des caractères alphanumériques, - (trait d'union) et _ (trait de soulignement). Il ne peut pas y avoir d'espaces.

S'il existe une incohérence de type entre un champ personnalisé dans votre configuration et la valeur en cours d'indexation, l'indexation de flotte ignore la valeur incohérente pour les requêtes d'agrégation. CloudWatch Les journaux sont utiles pour résoudre les problèmes liés aux requêtes d'agrégation. Pour de plus amples informations, veuillez consulter Dépannage des requêtes d'agrégation pour le service d'indexation de parc.

- Types de champs personnalisés

Les types de champs personnalisés ont les valeurs prises en charge suivantes : `Number`, `String`, et `Boolean`.

Gérer l'indexation d'objet

`AWS_Things` est l'index créé pour l'ensemble de vos objets. Vous pouvez contrôler les éléments à indexer à partir des sources de données suivantes : données de AWS IoT registre, données AWS IoT Device Shadow, données de AWS IoT connectivité et données de AWS IoT Device Defenderviolations.

Dans cette rubrique :

- Activation de l'indexation d'objet
- Description d'un index d'objets
- Interrogation d'un index d'objets
- Restrictions et limitations
- Autorisation

Activation de l'indexation d'objet

Vous utilisez la commande update-indexing-configurationCLI ou l'opération UpdateIndexingConfigurationAPI pour créer l'`AWS_Things`index et contrôler sa configuration. En utilisant le paramètre `--thing-indexing-configuration` (`thingIndexingConfiguration`),

vous contrôlez le type de données (par exemple, les données de registre, shadow, de connectivité des appareils et les données de violations de Device Defender) qui sont indexées.

Le paramètre `--thing-indexing-configuration` prend une chaîne avec la structure suivante :

```
{
  "thingIndexingMode": "OFF"|"REGISTRY"|"REGISTRY_AND_SHADOW",
  "thingConnectivityIndexingMode": "OFF"|"STATUS",
  "deviceDefenderIndexingMode": "OFF"|"VIOLATIONS",
  "namedShadowIndexingMode": "OFF"|"ON",
  "managedFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "customFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "filter": {
     "namedShadowNames": [ "string" ],
     "geoLocations": [
        {
            "name": "String",
            "order": "LonLat|LatLon"
        }
    ]
  }
}
```

Modes d'indexation d’objets

Vous pouvez définir différents modes d'indexation d'objets dans votre configuration d'indexation, en fonction des sources de données que vous souhaitez indexer et à partir desquelles vous souhaitez rechercher des appareils :

- `thingIndexingMode`: Contrôle si le registre ou shadow sont indexés. Lorsque `thingIndexingMode` est défini sur `OFF`, l'indexation des objets est désactivée.

- `thingConnectivityIndexingMode` : Indique si les données de connectivité d'objets sont indexées.

- `deviceDefenderIndexingMode` : Indique si les données relatives aux violations de Device Defender sont indexées.
- `namedShadowIndexingMode` : Indique si les données fictives nommées sont indexées. Pour sélectionner des shadows nommées à ajouter à la configuration d'indexation de votre flotte, configurez `namedShadowIndexingMode` comme étant `ON` et spécifiez les noms de vos shadows nommées dans `filter`.

Le tableau ci-dessous indique les valeurs valides pour chaque mode d'indexation et la source de données indexée pour chaque valeur.

Attribut	Valeurs valides	Registre	Shadow	Connectivité	Violations du DD	Shadow nommé
`thingIndexingMode`	OFF					
	REGISTRY	✓				
	REGISTRY_AND_SHADOW	✓	✓			
`thingConnectivityIndexingMode`	Non spécifié.					
	OFF					
	STATUS			✓		
`deviceDefenderIndexingMode`	Non spécifié.					
	OFF					
	VIOLATIONS				✓	

Attribut	Valeurs valides	Registre	Shadow	Connectivité	Violations du DD	Shadow nommé
`namedShadowIndexingMode`	Non spécifié.					
	OFF					
	ON					✓

Champs gérés et champs personnalisés

Champs gérés

Les champs gérés contiennent des données associées aux objets, aux groupes d'objets, aux ombres des appareils, à la connectivité des appareils et aux violations de Device Defender. AWS IoT définit le type de données dans les champs gérés. Vous spécifiez les valeurs de chaque champ géré lorsque vous créez un objet AWS IoT . Par exemple, les noms d'objets, les groupes d'objets et les descriptions d'objets sont tous des champs gérés. L'indexation de flotte indexe les champs gérés en fonction du mode d'indexation que vous spécifiez. Les champs gérés ne peuvent pas être modifiés ni s'afficher dans `customFields`.

Champs personnalisés

Vous pouvez agréger les attributs des objets, les données de Device Shadow et les données relatives aux violations de Device Defender en créant des champs personnalisés pour les indexer. L'attribut `customFields` est une liste de paires de noms de champs et de types de données. Vous pouvez effectuer des requêtes d'agrégation en fonction du type de données. Le mode d'indexation que vous choisissez et qui affecte les champs peut être spécifié dans `customFields`. Par exemple, si vous spécifiez le mode d'indexation `REGISTRY`, vous ne pouvez pas spécifier un champ personnalisé à partir d'une shadow d'objet. Vous pouvez utiliser la commande update-indexing-configurationCLI pour créer ou mettre à jour les champs personnalisés (voir un exemple de commande dans la section Mise à jour des exemples de configuration d'indexation). Pour plus d'informations, consultez Champs personnalisés.

Filtre d'indexation

Le filtre d'indexation fournit des sélections supplémentaires pour les shadows nommées et les données de géolocalisation.

`namedShadowNames`

Pour sélectionner des shadows nommées à ajouter à la configuration d'indexation de votre flotte, configurez `namedShadowIndexingMode` comme étant `ON` et spécifiez les noms de vos shadows nommées dans `namedShadowNames`.

Exemple

```
"filter": {
     "namedShadowNames": [ "namedShadow1", "namedShadow2" ]
}
```

`geoLocations`

Pour ajouter des données de géolocalisation à la configuration d'indexation de votre flotte :

- Si vos données de géolocalisation sont stockées dans une shadow classique (sans nom), définissez `thingIndexingMode` REGISTRY_AND_SHADOW et spécifiez vos données de géolocalisation dans le filtre `geoLocations`.

  L'exemple de filtre ci-dessous spécifie un objet GeoLocation dans une shadow classique (sans nom) :

  ```
  "filter": {
       "geoLocations": [
          {
              "name": "shadow.reported.location",
              "order": "LonLat"
          }
       ]
    }
  ```

- Si vos données de géolocalisation sont stockées dans une shadow nommée, `namedShadowIndexingMode` réglez-la sur ON, ajoutez le nom de shadow dans le filtre `namedShadowNames` et spécifiez vos données de géolocalisation dans le filtre `geoLocations`.

  L'exemple de filtre ci-dessous spécifie un objet geoLocation dans une shadow nommée (`nameShadow1`) :

  ```
  "filter": {
       "namedShadowNames": [ "namedShadow1" ],
       "geoLocations": [
  ```

```
        {
            "name": "shadow.name.namedShadow1.reported.location",
            "order": "LonLat"
        }
    ]
}
```

Pour plus d'informations, consultez le IndexingFilterGuide de référence des AWS IoTAPI.

Mise à jour des exemples de configuration d'indexation

Pour mettre à jour votre configuration d'indexation, utilisez la commande AWS IoT update-indexing-configuration CLI. Les exemples suivants montrent comment utiliser update-indexing-configuration.

Syntaxe courte :

```
aws iot update-indexing-configuration --thing-indexing-configuration \
'thingIndexingMode=REGISTRY_AND_SHADOW, deviceDefenderIndexingMode=VIOLATIONS,
namedShadowIndexingMode=ON,filter={namedShadowNames=[thing1shadow]},
 thingConnectivityIndexingMode=STATUS,
customFields=[{name=attributes.version,type=Number},
{name=shadow.name.thing1shadow.desired.DefaultDesired, type=String},
 {name=shadow.desired.power, type=Boolean},
{name=deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number,
 type=Number}]'
```

Syntaxe JSON :

```
aws iot update-indexing-configuration --cli-input-json \ '{
          "thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY_AND_SHADOW",
          "thingConnectivityIndexingMode": "STATUS",
          "deviceDefenderIndexingMode": "VIOLATIONS",
          "namedShadowIndexingMode": "ON",
          "filter": { "namedShadowNames": ["thing1shadow"]},
          "customFields": [ { "name": "shadow.desired.power", "type": "Boolean" },
          {"name": "attributes.version", "type": "Number"},
          {"name": "shadow.name.thing1shadow.desired.DefaultDesired", "type":
 "String"},
          {"name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
 "type": Number} ] } }'
```

Cette commande ne produit aucune sortie.

Pour vérifier l'état de l'index des objets, exécutez la commande CLI `describe-index` :

```
aws iot describe-index --index-name "AWS_Things"
```

La sortie de la commande `describe-index` ressemble à ce qui suit :

```
{
    "indexName": "AWS_Things",
    "indexStatus": "ACTIVE",
    "schema": "MULTI_INDEXING_MODE"
}
```

> **Note**
>
> L'indexation de flotte peut prendre un certain temps pour mettre à jour l'indice de flotte. Nous vous recommandons d'attendre que le `indexStatus` affiche ACTIVE avant de l'utiliser. Le champ de schéma peut contenir différentes valeurs en fonction des sources de données que vous avez configurées. Pour plus d'informations, consultez Description d’un index d'objet.

Pour obtenir les détails de configuration de l'indexation de votre appareil, exécutez la commande CLI `get-indexing-configuration` :

```
aws iot get-indexing-configuration
```

La sortie de la commande `get-indexing-configuration` ressemble à ce qui suit :

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY_AND_SHADOW",
        "thingConnectivityIndexingMode": "STATUS",
        "deviceDefenderIndexingMode": "VIOLATIONS",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "connectivity.disconnectReason",
                "type": "String"
            },
            {
```

```
            "name": "registry.version",
            "type": "Number"
        },
        {
            "name": "thingName",
            "type": "String"
        },
        {
            "name": "deviceDefender.violationCount",
            "type": "Number"
        },
        {
            "name": "shadow.hasDelta",
            "type": "Boolean"
        },
        {
            "name": "shadow.name.*.version",
            "type": "Number"
        },
        {
            "name": "shadow.version",
            "type": "Number"
        },
        {
            "name": "connectivity.version",
            "type": "Number"
        },
        {
            "name": "connectivity.timestamp",
            "type": "Number"
        },
        {
            "name": "shadow.name.*.hasDelta",
            "type": "Boolean"
        },
        {
            "name": "registry.thingTypeName",
            "type": "String"
        },
        {
            "name": "thingId",
            "type": "String"
        },
        {
```

```
                "name": "connectivity.connected",
                "type": "Boolean"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "shadow.name.thing1shadow.desired.DefaultDesired",
                "type": "String"
            },

            {
                "name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
                "type": "Number"
            },
            {
                "name": "shadow.desired.power",
                "type": "Boolean"
            },
            {
                "name": "attributes.version",
                "type": "Number"
            }
        ],
        "filter": {
              "namedShadowNames": [
                  "thing1shadow"
              ]
          }
      },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Pour mettre à jour les champs personnalisés, vous pouvez exécuter la commande `update-indexing-configuration`. Un exemple se présente comme suit :

```
aws iot update-indexing-configuration --thing-indexing-configuration
```

```
 'thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.version,type=Number},
{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean},
{name=shadow.desired.intensity,type=Number}]'
```

Cette commande a ajouté `shadow.desired.intensity` à la configuration d'indexation.

> **Note**
>
> La mise à jour de la configuration d'indexation des champs personnalisés remplace tous les champs personnalisés existants. Assurez-vous de spécifier tous les champs personnalisés lorsque vous appelez update-indexing-configuration.

Une fois l'index régénéré, vous pouvez utiliser une requête d'agrégation sur les champs nouvellement ajoutés et faire des recherches dans les données de registre, les données de shadow et les données de statut de connectivité d'objet.

Lorsque vous modifiez le mode d'indexation, assurez-vous que tous vos champs personnalisés sont valides à l'aide du nouveau mode d'indexation. Par exemple, si vous commencez par le mode `REGISTRY_AND_SHADOW` avec un champ personnalisé appelé, `shadow.desired.temperature`, vous devez supprimer le champ personnalisé `shadow.desired.temperature` avant de remplacer le mode d'indexation par `REGISTRY`. Si votre configuration d'indexation contient des champs personnalisés qui ne sont pas indexés par le mode d'indexation, la mise à jour échoue.

Description d'un index d'objets

La commande suivante montre comment utiliser la commande d'interface de ligne de commande describe-index pour extraire l'état actuel de l'index d'objets.

```
aws iot describe-index --index-name "AWS_Things"
```

La réponse de la commande peut ressembler à ce qui suit :

```
{
    "indexName": "AWS_Things",
    "indexStatus": "BUILDING",
    "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
}
```

La première fois que vous indexez une flotte, votre AWS IoT index est créé. Lorsque `indexStatus` est dans l'état `BUILDING`, vous ne pouvez pas interroger l'index. Le `schema` de l'index d'objets indique le type de données (`REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS`) qui est indexé.

Si vous modifiez la configuration de votre index, ce dernier est recréé. Lors de ce processus, l'`indexStatus` est `REBUILDING`. Vous pouvez exécuter des requêtes sur les données de l'index d'objets pendant sa régénération. Par exemple, si vous faites passer la configuration d'index de `REGISTRY` à `REGISTRY_AND_SHADOW`, pendant sa régénération, vous pouvez interroger les données de registre, y compris les dernières mises à jour. Toutefois, vous ne pouvez pas interroger les données des shadows tant que la reconstruction n'est pas terminée. Le temps nécessaire pour créer ou recréer l'index dépend de la quantité de données.

Vous pouvez voir différentes valeurs dans le champ de schéma en fonction des sources de données que vous avez configurées. Le tableau suivant affiche les différentes valeurs de schéma et les descriptions correspondantes :

Schema	Description
OFF	Aucune source de données n'est configurée ou indexée.
REGISTRY	Les données du registre sont indexées.
REGISTRY_AND_SHADOW	Les données du registre et les données shadow sans nom (classiques) sont indexées.
REGISTRY_ET_CONNECTIVITÉ_	Les données de registre et les données de connectivité sont indexées.
REGISTRY_AND_SHADOW_AND_CON NECTIVITY_STATUS REGISTRY_ AND_CONNECTIVITY_STATUS	Les données de registre, les données shadow sans nom (classiques) et les données de connectivité sont indexées.
MULTI_INDEXING_MODE	Les données relatives aux violations de Named Shadow ou Device Defender sont indexées, en plus des données de registre, shadow anonyme (classique) ou de connectivité.

Interrogation d'un index d'objets

Utilisez la commande de l'interface de ligne de commande search-index pour interroger les données dans l'index.

```
aws iot search-index --index-name "AWS_Things" --query-string
          "thingName:mything*"
```


```
{
    "things":[{
         "thingName":"mything1",
         "thingGroupNames":[
            "mygroup1"
         ],
         "thingId":"a4b9f759-b0f2-4857-8a4b-967745ed9f4e",
         "attributes":{
            "attribute1":"abc"
         },
         "connectivity": {
            "connected":false,
            "timestamp":1556649874716,
            "disconnectReason": "CONNECTION_LOST"
         }
    },
    {
        "thingName":"mything2",
        "thingTypeName":"MyThingType",
        "thingGroupNames":[
            "mygroup1",
            "mygroup2"
        ],
        "thingId":"01014ef9-e97e-44c6-985a-d0b06924f2af",
        "attributes":{
            "model":"1.2",
            "country":"usa"
        },
        "shadow":{
            "desired":{
                "location":"new york",
                "myvalues":[3, 4, 5]
            },
            "reported":{
                "location":"new york",
```

```
                "myvalues":[1, 2, 3],
                "stats":{
                    "battery":78
                }
            },
            "metadata":{
                 "desired":{
                       "location":{
                            "timestamp":123456789
                        },
                       "myvalues":{
                            "timestamp":123456789
                       }
                 },
                 "reported":{
                       "location":{
                            "timestamp":34535454
                        },
                       "myvalues":{
                            "timestamp":34535454
                       },
                       "stats":{
                            "battery":{
                                 "timestamp":34535454
                            }
                       }
                }
            },
            "version":10,
            "timestamp":34535454
        },
        "connectivity": {
            "connected":true,
            "timestamp":1556649855046
        }
    }],
    "nextToken":"AQFCuvk7zZ3D9pOYMbFCeHbdZ+h=G"
}
```

Dans la réponse JSON `"connectivity"` (telle qu'activée par le paramètre `thingConnectivityIndexingMode=STATUS`), fournit une valeur booléenne, un horodatage et un déconnecterReason qui indique si l'appareil est connecté à AWS IoT Core. L’appareil `"mything1"`

déconnecté (`false`) à l'heure POSIX `1556649874716` à cause de `CONNECTION_LOST`. Pour plus d'informations sur les motifs de déconnexion, consultez Événements du cycle de vie.

```
"connectivity": {
    "connected":false,
    "timestamp":1556649874716,
    "disconnectReason": "CONNECTION_LOST"
}
```

L'appareil `"mything2"` connecté (`true`) à l'heure POSIX `1556649855046` :

```
"connectivity": {
    "connected":true,
    "timestamp":1556649855046
}
```

Les horodatages sont indiqués en millisecondes depuis l'époque Unix, donc `1556649855046` représente 6:44:15.046 PM le mardi 30 avril 2019 (UTC).

> **Important**
>
> Si un appareil a été déconnecté pendant environ une heure, la valeur `"timestamp"` et la valeur `"disconnectReason"` de l'état de connectivité peuvent être manquantes.

Restrictions et limitations

Les restrictions et limitations pour `AWS_Things` sont les suivantes.

Champs Shadow de types complexes

Un champ shadow est indexé uniquement si la valeur du champ est un type simple, tel qu'un objet JSON qui ne contient pas de tableau ou un tableau qui se compose entièrement de types simples. (Un type simple représente une chaîne, un nombre ou un des littéraux `true` ou `false`.) Par exemple, étant donné l'état du shadow suivant, la valeur du champ `"palette"` n'est pas indexée, car il s'agit d'un tableau dont les éléments ont des types complexes. La valeur du champ `"colors"` est indexée, car chaque valeur du tableau est une chaîne.

```
{
    "state": {
```

```
        "reported": {
            "switched": "ON",
            "colors": [ "RED", "GREEN", "BLUE" ],
            "palette": [
                {
                    "name": "RED",
                    "intensity": 124
                },
                {
                    "name": "GREEN",
                    "intensity": 68
                },
                {
                    "name": "BLUE",
                    "intensity": 201
                }
            ]
        }
    }
}
```

Noms de champs shadow imbriqués

Les noms des champs shadow imbriqués sont stockés sous la forme d'une chaîne délimitée par un point (.). Par exemple, soit un document shadow :

```
{
  "state": {
    "desired": {
      "one": {
        "two": {
          "three": "v2"
        }
      }
    }
  }
}
```

Le nom du champ `three` est stocké sous la forme `desired.one.two.three`. Si vous disposez également d'un document shadow, il est stocké comme ceci :

```
{
  "state": {
```

```
    "desired": {
      "one.two.three": "v2"
    }
  }
}
```

Ils correspondent tous deux à une requête pour `shadow.desired.one.two.three:v2`. Conformément aux bonnes pratiques, n'utilisez pas de points dans les noms de champs shadow.

Métadonnées de shadow

Un champ d'une section de métadonnées de shadow est indexé, à l'unique condition que le champ correspondant dans la section `"state"` du shadow soit indexé. (Dans l'exemple précédent, le champ `"palette"` de la section des métadonnées du shadow n'est pas indexé non plus.)

Appareils non enregistrés

L'indexation de flotte indexe l'état de connectivité d'un appareil dont la connexion `clientId` est identique à celle `thingName` d'un objet enregistré dans le registre .

Shadows non enregistrés

Si vous créez une ombre en utilisant un nom d'objet qui n'a pas été enregistré dans votre AWS IoT compte, les champs de cette ombre ne sont pas indexés. UpdateThingShadow Cela s'applique à la fois à la shadow classique anonyme et nommée.

Valeur numériques

Si des données de registre ou de shadow sont reconnues par le service en tant que valeurs numériques, elles sont indexées en tant que telles. Vous pouvez formuler des requêtes comportant des plages et des opérateurs de comparaison sur les valeurs numériques (par exemple, `"attribute.foo<5"` ou `"shadow.reported.foo:[75 TO 80]"`). Pour être reconnue comme numérique, la valeur des données doit être un numéro JSON valide et littéral. La valeur peut être un entier compris entre -2 ^ 53...2 ^ 53-1, une virgule flottante double précision avec une notation exponentielle facultative ou une partie d'un tableau contenant uniquement ces valeurs.

Valeurs nulles

Les valeurs nulles ne sont pas indexées.

Valeurs maximales

Le nombre maximal de champs personnalisés pour les requêtes d'agrégation est 5.

Le nombre maximal de centiles demandés pour les requêtes d'agrégation est 100.

Autorisation

Vous pouvez spécifier l'index des objets sous la forme d'un Amazon Resource Name (ARN) dans le cadre d'une action AWS IoT politique, comme suit.

Action	Ressource
`iot:SearchIndex`	Un ARN d'index (par exemple, `arn:aws:iot:` *`your-aws-region your-aws-account`* `:index/AWS_Things`).
`iot:DescribeIndex`	Un ARN d'index (par exemple, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_Things`).

> **Note**
>
> Si vous disposez d'autorisations pour interroger l'index de la flotte, vous pouvez accéder aux données d'objets dans la totalité de la flotte.

Gérer l'indexation du groupe d'objet

`AWS_ThingGroups` est l'index qui contient tous les groupes de votre objet. Cet index vous permet de rechercher des groupes en fonction de leur nom, de la description, des attributs et de tous les noms de groupes parents.

Activation de l'indexation de groupes d'objets

Vous pouvez utiliser le `thing-group-indexing-configuration` paramètre de l'UpdateIndexingConfigurationAPI pour créer l'`AWS_ThingGroups`index et contrôler sa configuration. Vous pouvez utiliser l'GetIndexingConfigurationAPI pour récupérer la configuration d'indexation actuelle.

Pour mettre à jour les configurations d'indexation des groupes d'objets, exécutez la commande CLI update-indexing-configuration :

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

Vous pouvez également mettre à jour les configurations d'indexation d'objets et de groupes d'objets avec une seule commande, comme indiqué ci-dessous :

```
aws iot update-indexing-configuration --thing-indexing-configuration
 thingIndexingMode=REGISTRY --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

Les valeurs suivantes sont valides pour `thingGroupIndexingMode`.

OFF

Pas d' indexing/delete index.

ON

Créez ou configurez l'index `AWS_ThingGroups`.

Pour récupérer les configurations actuelles d'indexation des objets et des groupes d'objets, exécutez la commande CLI get-indexing-configuration :

```
aws iot get-indexing-configuration
```

La réponse de la commande ressemble à ce qui suit :

```
{
   "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "ON"
    }
}
```

Description des index de groupes

Pour récupérer l'état actuel de l'index `AWS_ThingGroups`, utilisez la commande CLI describe-index :

```
aws iot describe-index --index-name "AWS_ThingGroups"
```

La réponse de la commande ressemble à ce qui suit :

```
{
   "indexStatus": "ACTIVE",
   "indexName": "AWS_ThingGroups",
   "schema": "THING_GROUPS"
}
```

AWS IoT crée votre index lors de votre première indexation. Vous ne pouvez pas interroger l'index si le `indexStatus` est `BUILDING`.

Interrogation d'un index de groupes d'objets

Pour interroger les données de l'index, utilisez la commande CLI search-index :

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string
 "thingGroupName:mythinggroup*"
```

Autorisation

Vous pouvez spécifier l'index des groupes d'objets en tant qu'ARN de ressource dans le cadre d'une action de AWS IoT politique, comme suit.

Action	Ressource
`iot:SearchIndex`	Un ARN d'index (par exemple, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).
`iot:DescribeIndex`	Un ARN d'index (par exemple, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Demandes d'état de connectivité de l'appareil

AWS IoT Fleet Indexing prend en charge les requêtes de connectivité individuelles des appareils, ce qui vous permet de récupérer efficacement l'état de connectivité et les métadonnées associées pour des appareils spécifiques. Cette fonctionnalité complète les fonctionnalités d'indexation et d'interrogation existantes à l'échelle du parc.

Comment ça marche

La prise en charge des requêtes de connectivité des appareils peut être utilisée pour une récupération optimisée de l'état de connectivité d'un seul appareil. Cette API fournit un accès à faible latence et à haut débit aux informations de connectivité spécifiques aux appareils les plus récentes. Une fois que vous aurez activé l'indexation de connectivité, vous aurez accès à cette API de requête qui sera facturée en tant que requêtes standard. Pour plus d'informations, consultez la section Tarification de la gestion des AWS IoT appareils

Fonctionnalités

Grâce à la prise en charge des requêtes relatives à la connectivité des appareils, vous pouvez :

1. Recherchez l'état de connectivité actuel (connecté ou déconnecté) d'un appareil donné à l'aide de son`thingName`.
2. Récupérez des métadonnées de connectivité supplémentaires, notamment :
   a. Raison de la déconnexion
   b. Horodatage du dernier événement de connexion ou de déconnexion.

> **Note**
>
> L'indexation de flotte indexe l'état de connectivité d'un appareil dont la connexion `clientId` est identique à celle `thingName` d'un objet enregistré dans le registre .

Avantages

1. Faible latence : reflète l'état de connectivité le plus récent de l'appareil et offre une faible latence pour refléter les changements d'état de connexion depuis IoT Core. IoT Core détermine qu'un appareil est déconnecté soit dès qu'il reçoit une demande de déconnexion de la part de l'appareil, soit dans le cas où un appareil se déconnecte sans envoyer de demande de déconnexion.
   Le cœur de l'IoT attendra 1,5 fois le temps de maintien en vie configuré avant qu'il ne soit déterminé que le client est déconnecté. L'API d'état de la connectivité reflétera ces changements généralement moins d'une seconde après que IoT Core ait déterminé le changement d'état de connexion d'un appareil.
2. Haut débit : prend en charge 350 transactions par seconde (TPS) par défaut et peut être ajusté à un niveau supérieur sur demande.

3. Conservation des données : stocke les données des événements indéfiniment lorsque le ConnectivityIndexing mode Fleet Indexing (FI) est activé et que l'objet n'est pas supprimé. Si vous désactivez l'indexation de connectivité, les enregistrements ne seront pas conservés.

> **Note**
>
> Si l'indexation de l'état de connectivité a été activée avant le lancement de cette API, Fleet Indexing commence à suivre les modifications de l'état de connectivité après le lancement de l'API et reflète le statut mis à jour en fonction de ces modifications.

Prérequis

Pour utiliser la prise en charge des requêtes de connectivité de l'appareil :

1. Créez un AWS compte
2. Intégrez et enregistrez les appareils AWS IoT Core dans votre région préférée
3. Activez l'indexation de la flotte grâce à l'indexation de la connectivité

> **Note**
>
> Aucune configuration supplémentaire n'est requise si l'indexation de connectivité est déjà activée

Pour des instructions de configuration détaillées, reportez-vous au guide du AWS IoT développeur

Exemples

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": true,
   "disconnectReason": "NONE",
   "thingName": "myThingName",
```

```
   "timestamp": "2024-12-19T10:00:00.000000-08:00"
}
```

- `thingName`: nom de l'appareil tel qu'indiqué dans la demande. Cela correspond également au ClientiD utilisé pour se connecter. AWS IoT Core
- `disconnectReason`: Motif de la déconnexion. Ce sera AUCUN pour un appareil connecté.
- `connected`: valeur booléenne vraie indiquant que cet appareil est actuellement connecté.
- `timestamp`: horodatage représentant la dernière déconnexion de l'appareil en millisecondes.

```
aws iot get-thing-connectivity-data --thing-name myThingName
```

```
{
   "connected": false,
   "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
   "thingName": "myThingName",
   "timestamp": "2024-12-19T10:30:00.000000-08:00"
}
```

- `thingName`: nom de l'appareil tel qu'indiqué dans la demande. Cela correspond également au ClientiD utilisé pour se connecter. AWS IoT Core
- `disconnectReason`: La raison de la déconnexion est CLIENT_INITIATED_DISCONNECT, qui indique que le client a indiqué qu'il allait se AWS IoT Core déconnecter.
- `connected`: valeur booléenne fausse indiquant que cet appareil est actuellement déconnecté.
- `timestamp`: horodatage représentant la dernière déconnexion de l'appareil en millisecondes.

```
aws iot get-thing-connectivity-data --thing-name neverConnectedThing
```

```
{
   "connected": false,
   "disconnectReason": "UNKNOWN",
   "thingName": "neverConnectedThing"
}
```

- `thingName`: nom de l'appareil tel qu'indiqué dans la demande. Cela correspond également au ClientiD utilisé pour se connecter. AWS IoT Core
- `disconnectReason`: Motif de la déconnexion. Sera « INCONNU » pour un appareil qui n'a jamais été connecté ou pour lequel Fleet Indexing n'a pas enregistré la dernière raison de déconnexion.
- `connected`: valeur booléenne fausse indiquant que cet appareil est actuellement déconnecté.
- `timestamp`: L'horodatage n'est pas renvoyé pour un appareil qui n'a jamais été connecté ou pour lequel le dernier horodatage n'est pas enregistré dans Fleet Indexing.

Interrogation des données agrégées

AWS IoT en fournit quatre APIs (`GetStatistics`, `GetCardinalityGetPercentiles`, et`GetBucketsAggregation`) qui vous permettent de rechercher des données agrégées dans votre parc d'appareils.

> **Note**
>
> Pour les problèmes liés à des valeurs manquantes ou inattendues pour l'agrégation APIs, consultez le guide de résolution des problèmes liés à l'indexation de la flotte.

GetStatistics

L'GetStatisticsAPI et la commande get-statistics CLI renvoient le nombre, la moyenne, la somme, le minimum, le maximum, la somme des carrés, la variance et l'écart type pour le champ agrégé spécifié.

La commande get-statistics de l'interface de ligne de commande (CLI) utilise les paramètres suivants :

`index-name`

Nom de l'index dans lequel effectuer la recherche. La valeur par défaut est `AWS_Things`.

`query-string`

Nom de la requête utilisée pour recherche dans l'index. Vous pouvez spécifier `"*"` d'obtenir le nombre de tous les éléments indexés dans votre Compte AWS.

`aggregationField`

(Facultatif) Le champ à agréger. Ce champ doit être un champ géré ou personnalisé défini lorsque vous appelez update-indexing-configuration. Si vous ne spécifiez pas de champ d'agrégation, `registry.version` est utilisé comme le champ d'agrégation.

`query-version`

Version de la requête à utiliser. La valeur par défaut est `2017-09-30`.

Le type de champ d'agrégation peut affecter les statistiques renvoyées.

GetStatistics avec des valeurs de chaîne

Si vous regroupez les données en fonction d'un champ de chaîne, l'appel à `GetStatistics` renvoie un nombre d'appareils dont les attributs correspondent à la requête. Par exemple :

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute'
            --query-string '*'
```

Cette commande renvoie le nombre d'appareils qui contiennent un attribut nommé `stringAttribute` :

```
{
  "statistics": {
    "count": 3
  }
}
```

GetStatistics avec des valeurs booléennes

Lorsque vous appelez `GetStatistics` avec un champ d'agrégation Booléen :

- AVERAGE est le pourcentage d'appareils qui correspondent à la requête.
- MINIMUM est 0 ou 1, d'après les règles suivantes :
  - Si toutes les valeurs du champ d'agrégation sont `false`, MINIMUM est 0.
  - Si toutes les valeurs du champ d'agrégation sont `true`, MINIMUM est 1.
  - Si des valeurs du champ d'agrégation sont `false` et d'autres `true`, MINIMUM est 0.
- MAXIMUM est 0 ou 1, d'après les règles suivantes :

  - Si toutes les valeurs du champ d'agrégation sont `false`, MAXIMUM est 0.
  - Si toutes les valeurs du champ d'agrégation sont `true`, MAXIMUM est 1.
  - Si des valeurs du champ d'agrégation sont `false` et d'autres `true`, MAXIMUM est 1.
- SUM est la somme de l'équivalent entier des valeurs Booléennes.
- COUNT est le nombre d'éléments qui correspondent aux critères de chaîne de requête et contiennent une valeur de champ d'agrégation valide.

GetStatistics avec des valeurs numériques

Lorsque vous appelez `GetStatistics` et spécifiez un champ d'agrégation de type `Number`, `GetStatistics` renvoie les valeurs suivantes :

count

Nombre d'éléments qui correspondent aux critères de la chaîne de requête et contiennent une valeur de champ d'agrégation valide.

average

Moyenne des valeurs numériques qui correspondent à la requête.

sum

Somme des valeurs numériques qui correspondent à la requête.

minimum

La plus petite des valeurs numériques qui correspondent à la requête.

maximum

La plus grande des valeurs numériques qui correspondent à la requête.

sumOfSquares

Somme des carrés des valeurs numériques qui correspondent à la requête.

variance

Variance des valeurs numériques qui correspondent à la requête. La variance d'un ensemble de valeurs est la moyenne des carrés des différences de chaque valeur par rapport à la valeur moyenne de l'ensemble.

stdDeviation

Écart type des valeurs numériques qui correspondent à la requête. L'écart type d'un ensemble de valeurs est une mesure de la répartition des valeurs.

L'exemple suivant montre comment appeler get-statistics avec un champ numérique personnalisé.

```
aws iot get-statistics --aggregation-field 'attributes.numericAttribute2'
            --query-string '*'
```

```
{
  "statistics": {
    "count": 3,
    "average": 33.333333333333336,
    "sum": 100.0,
    "minimum": -125.0,
    "maximum": 150.0,
    "sumOfSquares": 43750.0,
    "variance": 13472.22222222222,
    "stdDeviation": 116.06990230986766
  }
}
```

crée votre index la première fois que vous indexez.

GetCardinality

L'GetCardinalityAPI et la commande get-cardinality CLI renvoient le nombre approximatif de valeurs uniques correspondant à la requête. Par exemple, vous pouvez trouver le nombre d'appareils dont le niveau de batterie est inférieur à 50 % :

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel
          > 50" --aggregation-field "shadow.reported.batterylevel"
```

Cette commande renvoie le nombre d'objets dont le niveau de batterie est supérieur à 50 % :

```
{
    "cardinality": 100
}
```

`cardinality` est toujours renvoyé par get-cardinality, même s'il n'y a pas de champs correspondants. Par exemple :

```
aws iot get-cardinality --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_STR"
```

```
{
    "cardinality": 0
}
```

La commande get-cardinality de l'interface de ligne de commande (CLI) utilise les paramètres suivants :

`index-name`

Nom de l'index dans lequel effectuer la recherche. La valeur par défaut est `AWS_Things`.

`query-string`

Nom de la requête utilisée pour recherche dans l'index. Vous pouvez spécifier `"*"` d'obtenir le nombre de tous les éléments indexés dans votre Compte AWS.

`aggregationField`

Champ à agréger.

`query-version`

Version de la requête à utiliser. La valeur par défaut est `2017-09-30`.

GetPercentiles

L'GetPercentilesAPI et la commande get-percentiles CLI regroupent les valeurs agrégées correspondant à la requête en groupes percentiles. Les groupes de centiles par défaut sont : 1,5,25,50,75,95,99, bien que vous puissiez spécifier les vôtres lorsque vous appelez `GetPercentiles`. Cette fonction renvoie une valeur pour chaque groupe de centiles spécifié (ou les groupes de centiles par défaut). Le groupe de centiles « 1 » contient la valeur de champ agrégée qui se produit dans environ 1 % des valeurs qui correspondent à la requête. Le groupe de centiles « 5 » contient la valeur de champ agrégée qui se produit dans environ 5 % des valeurs qui correspondent à la requête, etc. Le résultat est une approximation. Plus les valeurs correspondent à la requête, plus les valeurs de centile sont précises.

L'exemple suivant montre comment appeler la commande de l'interface de ligne de commande get-percentiles.

```
aws iot get-percentiles --query-string "thingName:*" --aggregation-field
          "attributes.customField_NUM" --percents 10 20 30 40 50 60 70 80 90 99
```


```
{
    "percentiles": [
        {
            "value": 3.0,
            "percent": 80.0
        },
        {
            "value": 2.5999999999999996,
            "percent": 70.0
        },
        {
            "value": 3.0,
            "percent": 90.0
        },
        {
            "value": 2.0,
            "percent": 50.0
        },
        {
            "value": 2.0,
            "percent": 60.0
        },
        {
            "value": 1.0,
            "percent": 10.0
        },
        {
            "value": 2.0,
            "percent": 40.0
        },
        {
            "value": 1.0,
            "percent": 20.0
        },
        {
            "value": 1.4,
            "percent": 30.0
```

```
        },
        {
            "value": 3.0,
            "percent": 99.0
        }
    ]
}
```

La commande suivante affiche la sortie renvoyée par get-percentiles lorsqu'il n'y a pas de documents correspondants.

```
aws iot get-percentiles --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_NUM"
```

```
{
    "percentiles": []
}
```

La commande get-percentile de l'interface de ligne de commande (CLI) utilise les paramètres suivants :

`index-name`

Nom de l'index dans lequel effectuer la recherche. La valeur par défaut est `AWS_Things`.

`query-string`

Nom de la requête utilisée pour recherche dans l'index. Vous pouvez spécifier `"*"` d'obtenir le nombre de tous les éléments indexés dans votre Compte AWS.

`aggregationField`

Champ à agréger, dont le type doit être `Number`.

`query-version`

Version de la requête à utiliser. La valeur par défaut est `2017-09-30`.

`percents`

(Facultatif) Vous pouvez utiliser ce paramètre pour spécifier des regroupements de centiles personnalisés.

GetBucketsAggregation

L'GetBucketsAggregationAPI et la commande get-buckets-aggregation CLI renvoient une liste de compartiments et le nombre total d'éléments correspondant aux critères de chaîne de requête.

L'exemple suivant montre comment appeler la commande de l'interface de ligne de commande get-buckets-aggregation.

```
aws iot get-buckets-aggregation --query-string '*' --index-name AWS_Things --
aggregation-field 'shadow.reported.batterylevelpercent' --buckets-aggregation-type
 'termsAggregation={maxBuckets=5}'
```

Cette commande renvoie ce qui suit :

```
{
    "totalCount": 20,
    "buckets": [
        {
            "keyValue": "100",
            "count": 12
        },
        {
            "keyValue": "90",
            "count": 5
        },
        {
            "keyValue": "75",
            "count": 3
        }
    ]
}
```

La commande get-buckets-aggregation de l'interface de ligne de commande (CLI) utilise les paramètres suivants :

`index-name`

Nom de l'index dans lequel effectuer la recherche. La valeur par défaut est `AWS_Things`.

`query-string`

Nom de la requête utilisée pour recherche dans l'index. Vous pouvez spécifier `"*"` d'obtenir le nombre de tous les éléments indexés dans votre Compte AWS.

`aggregation-field`

Champ à agréger.

`buckets-aggregation-type`

Contrôle de base de la forme de réponse et du type d'agrégation de compartiments à effectuer.

Autorisation

Vous pouvez spécifier l'index des groupes d'objets en tant qu'ARN de ressource dans le cadre d'une action de AWS IoT politique, comme suit.

Action	Ressource
`iot:GetStatistics`	Un ARN d'index (par exemple, `arn:aws:iot:` *`your-aws-region`* `:index/AWS_Things` ou `arn:aws:iot:` *`your-aws-region`* `:index/AWS_ThingGroups`).

Syntaxe de requête

Dans le cadre de l'indexation de flotte, vous utilisez une syntaxe de requête pour spécifier les requêtes.

Fonctionnalités prises en charge

Cette syntaxe de requête prend en charge les fonctions ci-dessous :

- Termes et expressions
- Champs de recherche
- Recherche de préfixe
- Recherche de plage
- Opérateurs booléennes `AND`, `OR`, `NOT` et `-`. Le trait d'union est utilisé pour exclure quelque chose des résultats de recherche (par exemple, `thingName:(tv* AND -plasma)`).
- Regroupement
- Regroupement de champs

- Échappement de caractères spéciaux (comme avec \)

Fonctions non prises en charge

Cette syntaxe de requête ne prend pas en charge les fonctions suivantes :

- Recherche avec caractère générique en préfixe (par exemple, « *xyz »), mais la recherche de « * » donne un résultat de recherche contenant tous les objets
- Expressions régulières
- Promotion
- Classement
- Recherches approximatives
- Recherche de proximité
- Tri
- Agrégation
- Caractères spéciaux : `@,#,%,\,/,',;, et ,. Notez que , n'est pris en charge que dans les géorequêtes.

Remarques

Quelques remarques concernant le langage de requête :

- L'opérateur par défaut est AND. Une requête pour "thingName:abc thingType:xyz" équivaut à "thingName:abc AND thingType:xyz".
- Si aucun champ n'est spécifié, AWS IoT recherche le terme dans tous les champs Registry, Device Shadow et Device Defender.
- Tous les noms de champs sont sensibles à la casse.
- La recherche est insensible à la casse. Les mots sont séparés par des caractères d'espace vide, comme défini par l'élément Java Character.isWhitespace(int).
- L'indexation des données Device Shadow (shadows sans nom et nommées) comprend les sections rapportées, souhaitées, delta et de métadonnées.
- Il n'est pas possible d'effectuer une recherche sur les versions du registre et des shadows d'appareil, mais elle sont présentes dans la réponse.
- Le nombre maximum de termes dans une requête est de douze.

- Le caractère spécial `,` n'est pris en charge que dans les géorequêtes.

Exemples de requêtes sur des objets

Spécifiez les requêtes dans une chaîne de requête à l'aide d'une syntaxe de requête. Les requêtes sont transmises à l'API `SearchIndex`. Le tableau ci-après répertorie quelques exemples de chaînes de requête.

Chaîne de requête	Résultat
`abc`	Requêtes pour « abc » dans n'importe quel registre, dans n'importe quel champ shadow (shadow anonyme classique et nommée) ou dans n'importe quel champ de violations de Device Defender.
`thingName:myThingName`	Requêtes pour un objet portant le nom « myThingName ».
`thingName:my*`	Requêtes concernant les objets dont le nom commence par « my ».
`thingName:ab?`	Requêtes concernant les objets dont le nom contient la chaîne « ab » suivie d'un caractère supplémentaire, par exemple : « aba », « abb », « abc », etc.
`thingTypeName:aa`	Requêtes pour les objets qui sont associés au type « aa ».
`thingGroupNames:a`	Requêtes portant sur des objets dont le nom de groupe d'objets parent ou de groupe de facturation est « a ».
`thingGroupNames:a*`	Requêtes portant sur des objets dont le nom du groupe d'objets parent ou du groupe de facturation correspond au modèle « a* ».
`attributes.myAttribute:75`	Requêtes concernant les objets qui comportent un attribut nommé « myAttribute » ayant pour valeur 75.

Chaîne de requête	Résultat
`attributes.myAttribute: [75 TO 80]`	Requêtes pour des éléments avec un attribut nommé « myAttribute » dont la valeur se situe dans une plage numérique (75 à 80 inclus).
`attributes.myAttribute: {75 TO 80]`	Requêtes pour les éléments avec un attribut nommé « myAttribute » dont la valeur se situe dans la plage numérique (>75 et <=80).
`attributes.serialNumber: ["abcd" TO "abcf"]`	Requêtes pour des éléments avec un attribut nommé « serialNumber » dont la valeur se trouve dans une plage de chaînes alphanumériques. Cette requête renvoie les objets dont l'attribut « serialNumber » a la valeur « abcd », « abce » ou « abcf ».
`attributes.myAttri bute:i*t`	Requêtes pour des éléments avec un attribut nommé « myAttribute » où la valeur est « i », suivi d'un nombre quelconque de caractères, suivi de « t ».
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Requêtes concernant les objets qui combinent des termes en utilisant des expressions booléennes. Cette requête renvoie les objets qui comportent un attribut nommé « attr1 » de valeur « abc », un attribut nommé « attr2 » qui est inférieur à 5 et un attribut nommé « attr3 » qui n'est pas supérieur à 10.
`shadow.hasDelta:true`	Requêtes pour des éléments avec une shadow sans nom comportant un élément delta.
`NOT attributes.model:l egacy`	Requêtes concernant les objets dont l'attribut nommé « model » n'est pas défini sur « legacy ».

Chaîne de requête	Résultat
`shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy`	Requêtes concernant les objets possédant les caractéristiques suivantes : • L'attribut `stats.battery` du shadow de l'objet possède une valeur comprise entre 70 et 100. • Le texte « v2 » ou « v3 » se retrouve dans le nom, le nom de type ou les valeurs d'attribut de l'objet. • L'attribut `model` de l'objet n'est pas défini sur « legacy ».
`shadow.reported.myvalues:2`	Requêtes concernant les objets dont la plage `myvalues` dans la section reported du shadow contient une valeur 2.
`shadow.reported.location:* NOT shadow.desired.stats.battery:*`	Requêtes concernant les objets possédant les caractéristiques suivantes : • L'attribut `location` figure dans la section `reported` du shadow. • L'attribut `stats.battery` ne figure pas dans la section `desired` du shadow.
`shadow.name.<shadowName>.hasDelta:true`	Requêtes pour les objets qui ont une shadow avec le nom donné et également un élément delta.
`shadow.name.<shadowName>.desired.filament:*`	Requêtes pour les éléments qui ont une shadow avec le nom donné et également une propriété de filament souhaitée.
`shadow.name.<shadowName>.reported.location:*`	Requête les objets dotés d'une shadow portant le nom donné et dont l'attribut `location` existe dans la section signalée de la shadow nommée.
`connectivity.connected:true`	Requêtes pour tous les appareils connectés.
`connectivity.connected:false`	Requêtes pour tous les appareils déconnectés.

Chaîne de requête	Résultat
`connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Requêtes pour tous les appareils connectés avec un horodatage de connexion >= 1557651600000 and <= 1557867600000. Les horodatages sont indiqués en millisecondes depuis l'époque Unix.
`connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]`	Requêtes pour tous les appareils déconnectés avec un horodatage de déconnexion >= 1557651600000 and <= 1557867600000. Les horodatages sont indiqués en millisecondes depuis l'époque Unix.
`connectivity.connected:true AND connectivity.timestamp > 1557651600000`	Requêtes pour tous les appareils connectés avec un horodatage de connexion > 1557651600000. Les horodatages sont indiqués en millisecondes depuis l'époque Unix.
`connectivity.connected:*`	Requêtes pour tous les appareils comportant des informations de connectivité.
`connectivity.disconnectReason:*`	Requêtes pour tous les appareils avec connectivité DisconnectReason présente.
`connectivity.disconnectReason:CLIENT_INITIATED_DISCONNECT`	Requêtes pour tous les appareils déconnectés en raison de CLIENT_INITIATED_DISCONNECT.
`deviceDefender.violationCount:[0 TO 100]`	Les requêtes portant sur des objets présentant une valeur de nombre de violations de Device Defender comprise dans la plage numérique (0 à 100, inclus).
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.inViolation:true`	Requêtes concernant des éléments contraires au comportement `disconnectBehavior` défini dans le profil de sécurité `device-SecurityProfile`. Notez que Inviolation:false n'est pas une requête valide.

Chaîne de requête	Résultat
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationValue.number>2`	Requêtes concernant les éléments qui constituent une violation du comportement `disconnectBehavior` tel que défini dans le périphérique doté du profil de sécurité, SecurityProfile avec une valeur d'événement de dernière violation supérieure à 2.
`deviceDefender.<device-SecurityProfile>.disconnectBehavior.lastViolationTime>1634227200000`	Requêtes concernant les éléments qui constituent une violation du comportement `disconnectBehavior` tel que défini dans le périphérique doté du profil de sécurité, SecurityProfile avec un dernier événement de violation après une période spécifiée.
`shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Requêtes concernant des objets situés à une distance radiale de 15,5 km des coordonnées 47.6204, -122.3491 . Cette chaîne de requête s'applique lorsque vos données de localisation sont stockées dans une shadow nommée.
`shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`	Requêtes concernant des objets situés à une distance radiale de 15,5 km des coordonnées 47.6204, -122.3491 . Cette chaîne de requête s'applique lorsque vos données de localisation sont stockées dans un shadow classique.

Exemples de requêtes sur des groupes d'objets

Les requêtes sont spécifiées dans une chaîne de requête à l'aide d'une syntaxe de requête, puis transmises à l'API `SearchIndex`. Le tableau ci-après répertorie quelques exemples de chaînes de requête.

Chaîne de requête	Résultat
`abc`	Requêtes pour « abc » dans n'importe quel champ.
`thingGroupName:myGroupThingName`	Requêtes pour un groupe d'objets nommé « myGroupTh ing Nom ».

Chaîne de requête	Résultat
`thingGroupName:my*`	Requêtes concernant les groupes d'objets dont le nom commence par « my ».
`thingGroupName:ab?`	Requêtes pour les groupes d'objets dont les noms comportent « ab » plus un caractère supplémentaire (par exemple : « aba », « abb », « abc », etc.).
`attributes.myAttribute:75`	Requêtes concernant les groupes d'objets qui comporten t un attribut nommé « myAttribute » ayant pour valeur 75.
`attributes.myAttribute:[75 TO 80]`	Requêtes pour les groupes d'objets avec un attribut nommé « myAttribute » dont la valeur se situe dans une plage numérique (75 à 80 inclus).
`attributes.myAttribute:[75 TO 80]`	Requêtes concernant les groupes d'objets qui comporten t un attribut nommé « myAttribute », dont la valeur est comprise dans la plage numérique >75 et <=80.
`attributes.myAttribute: ["abcd" TO "abcf"]`	Requêtes concernant les groupes d'objets qui comporten t un attribut nommé « myAttribute », dont la valeur est comprise dans une plage de chaînes alphanumériques. Cette requête renvoie les groupes d'objets dont l'attribu t « serialNumber » a la valeur « abcd », « abce » ou « abcf ».
`attributes.myAttribute:i*t`	Requêtes concernant les groupes d'objets qui comporten t un attribut nommé « myAttribute » dont la valeur est « i », suivi de n'importe quel nombre de caractères, puis du caractère « t ».
`attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10`	Requêtes concernant les groupes d'objets qui combinent des termes en utilisant des expressions booléennes. Cette requête renvoie les groupes d'objets qui comporten t un attribut nommé « attr1 » de valeur « abc », un attribut nommé « attr2 » qui est inférieur à 5 et un attribut nommé « attr3 » qui n'est pas supérieur à 10.

Chaîne de requête	Résultat
`NOT attributes.myAttribute:cde`	Requêtes concernant les groupes d'objets dont l'attribut nommé « myAttribute » n'est pas « cde ».
`parentGroupNames:(myParentThingGroupName)`	Requêtes pour les groupes d'objets dont le nom du groupe parent correspond à « myParentThing GroupName ».
`parentGroupNames:(myParentThingGroupName OR myRootThingGroupName)`	Requêtes pour les groupes d'objets dont le nom du groupe parent correspond à myParentThing GroupName « » ou « myRootThing GroupName ».
`parentGroupNames:(myParentThingGroupNa*)`	Requêtes pour les groupes d'objets dont le nom du groupe parent commence par « myParentThing GroupNa ».

Indexation des données de localisation

Vous pouvez utiliser l'indexation de flotte AWS IoT pour indexer les dernières données de localisation envoyées par vos appareils et rechercher des appareils à l'aide de géorequêtes. Cette fonctionnalité résout les cas d'utilisation de surveillance et de gestion des appareils tels que le suivi de localisation et la recherche de proximité. L'indexation des emplacements fonctionne de la même manière que les autres fonctionnalités d'indexation de flotte, avec des configurations supplémentaires à spécifier dans l'indexation de vos objets.

Les cas d'utilisation courants incluent : rechercher et regrouper des appareils situés dans les limites géographiques souhaitées, obtenir des informations spécifiques à l'emplacement à l'aide de termes de requête liés aux métadonnées et à l'état de l'appareil à partir de sources de données indexées, fournir une vue granulaire telle que le filtrage des résultats sur une zone géographique spécifique pour réduire les délais de rendu dans les cartes de surveillance de votre flotte, suivre l'emplacement du dernier appareil signalé, identifier les appareils qui se trouvent en dehors des limites souhaitées et générer des alarmes à l'aide des métriques de la flotte. Pour commencer à utiliser l'indexation des emplacements et les géorequêtes, consultez ???.

Formats de données pris en charge

AWS IoT l'indexation de flotte prend en charge les formats de données de localisation suivants :

1. Représentation textuelle bien connue de systèmes de coordonnées de référence

   Une chaîne qui suit le format Informations géographiques - Représentation textuelle connue des systèmes de référence de coordonnées. Un exemple peut être `"POINT(long lat)"`.

2. Une chaîne qui représente les coordonnées

   Une chaîne au format `"latitude, longitude"` ou `"longitude, latitude"`. Si vous utilisez `"longitude, latitude"`, vous devez également spécifier `order` dans `geoLocations`. Un exemple peut être `"41.12,-71.34"`.

3. Un objet des clés lat(latitude), lon(longitude)

   Ce format est applicable à la shadow classique et nommée. Clés prises en charge : `lat`, `latitude`, `lon`, `long`, `longitude`. Un exemple peut être `{"lat": 41.12, "lon": -71.34}`.

4. Un tableau qui représente les coordonnées

   Un tableau au format `[lat,lon]` ou`[lon,lat]`. Si vous utilisez le format `[lon,lat]`, qui est identique aux coordonnées dans GeoJSON (applicable à la shadow classique et nommée), vous devez également spécifier `order` dans `geoLocations`.

   Un exemple peut être :

```
{
  "location": {
    "coordinates": [
      **Longitude**,
      **Latitude**
    ],
    "type": "Point",
    "properties": {
      "country": "United States",
      "city": "New York",
      "postalCode": "*****",
      "horizontalAccuracy": 20,
      "horizontalConfidenceLevel": 0.67,
      "state": "New York",
      "timestamp": "2023-01-04T20:59:13.024Z"
    }
  }
}
```

Comment indexer les données de localisation

Les étapes suivantes montrent comment mettre à jour la configuration d'indexation pour vos données de localisation et utiliser des géorequêtes pour rechercher des appareils.

1. Sachez où sont stockées vos données de localisation

   L'indexation des flottes prend actuellement en charge l'indexation des données de localisation stockées dans des shadows classiques ou nommées.
2. Utiliser les formats de données de localisation pris en charge

   Assurez-vous que le format de vos données de localisation suit l'un des formats de données pris en charge.
3. Mettre à jour la configuration de l'indexation

   Au minimum, activez la configuration d'indexation des objets (registre). Vous devez également activer l'indexation sur les shadows classiques ou nommées contenant vos données de localisation. Lorsque vous mettez à jour l'indexation de vos objets, vous devez inclure vos données de localisation dans la configuration d'indexation.
4. Créer et exécuter des géorequêtes

   En fonction de vos cas d'utilisation, créez des géorequêtes et exécutez-les pour rechercher des appareils. La géorequête que vous composez doit suivre la Syntaxe de requête. Vous trouverez quelques exemples dans ???.

Mettre à jour la configuration de l'indexation des objets

Pour indexer les données de localisation, vous devez mettre à jour la configuration d'indexation et inclure vos données de localisation. En fonction de l'endroit où vos données de localisation sont stockées, suivez les étapes pour mettre à jour votre configuration d'indexation :

Données de localisation stockées dans des shadows classiques

Si vos données de localisation sont stockées dans une shadow classique, vous devez définir `thingIndexingMode` comme tel `REGISTRY_AND_SHADOW` et spécifier vos données de localisation dans les champs `geoLocations` (`name`et `order`) de `filter`.

Dans l'exemple de configuration d'indexation d'objets suivant, vous spécifiez le chemin des données de localisation `shadow.reported.coordinates` comme `name` et `LonLat` comme `order`.

```
{
 "thingIndexingMode": "REGISTRY_AND_SHADOW",
 "filter": {
  "geoLocations": [
   {
    "name": "shadow.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  Le mode d'indexation contrôle si le registre ou le shadow est indexé. Lorsque `thingIndexingMode` est défini sur `OFF`, l'indexation des objets est désactivée.

  Pour indexer les données de localisation stockées dans un shadow classique, vous devez définir `thingIndexingMode` sur `REGISTRY_AND_SHADOW`. Pour de plus amples informations, veuillez consulter ???.

- `filter`

  Le filtre d'indexation fournit des sélections supplémentaires pour les shadows nommées et les données de géolocalisation. Pour de plus amples informations, veuillez consulter ???.

- `geoLocations`

  Liste des cibles de géolocalisation que vous sélectionnez pour indexer. Par défaut, le nombre maximal de cibles de géolocalisation pour l'indexation est 1. Pour augmenter la limite, consultez AWS IoT Device Management Quotas.

- `name`

  Nom du champ cible de géolocalisation. Un exemple de valeur de `name` peut être le chemin des données de localisation de votre shadow : `shadow.reported.coordinates`.

- `order`

  L'ordre du champ cible de géolocalisation. Valeurs valides : `LatLon` et `LonLat`. `LatLon` signifie latitude et longitude. `LonLat`signifie longitude et latitude. Ce champ est facultatif. La valeur par défaut est `LatLon`.

Données de localisation stockées dans des shadows nommées

Si vos données de localisation sont stockées dans une shadow nommée, définissez `namedShadowIndexingMode` comme étant `ON`, ajoutez le ou les noms de votre shadow nommée au champ `namedShadowNames` dans `filter` et spécifiez le chemin de vos données de position dans le champ `geoLocations` dans `filter`.

Dans l'exemple de configuration d'indexation d'objets suivant, vous spécifiez le chemin des données de localisation `shadow.name.namedShadow1.reported.coordinates` comme `name` et `LonLat` comme `order`.

```
{
 "thingIndexingMode": "REGISTRY",
 "namedShadowIndexingMode": "ON",
 "filter": {
  "namedShadowNames": [
   "namedShadow1"
  ],
  "geoLocations": [
   {
    "name": "shadow.name.namedShadow1.reported.coordinates",
    "order": "LonLat"
   }
  ]
 }
}
```

- `thingIndexingMode`

  Le mode d'indexation contrôle si le registre ou le shadow est indexé. Lorsque `thingIndexingMode` est défini sur `OFF`, l'indexation des objets est désactivée.

  Pour indexer les données de localisation stockées dans une shadow nommée, vous devez définir `thingIndexingMode` comme étant `REGISTRY` (ou`REGISTRY_AND_SHADOW`). Pour de plus amples informations, veuillez consulter ???.

- `filter`

  Le filtre d'indexation fournit des sélections supplémentaires pour les shadows nommées et les données de géolocalisation. Pour de plus amples informations, veuillez consulter ???.

- `geoLocations`

Liste des cibles de géolocalisation que vous sélectionnez pour indexer. Par défaut, le nombre maximal de cibles de géolocalisation pour l'indexation est 1. Pour augmenter la limite, consultez AWS IoT Device Management Quotas.

- `name`

  Nom du champ cible de géolocalisation. Un exemple de valeur de `name` peut être le chemin des données de localisation de votre shadow : `shadow.name.namedShadow1.reported.coordinates`.

- `order`

  L'ordre du champ cible de géolocalisation. Valeurs valides : `LatLon` et `LonLat`. `LatLon` signifie latitude et longitude. `LonLat`signifie longitude et latitude. Ce champ est facultatif. La valeur par défaut est `LatLon`.

Exemples de géorequêtes

Une fois que vous avez terminé la configuration d'indexation de vos données de localisation, exécutez des géorequêtes pour rechercher des appareils. Vous pouvez également combiner vos géorequêtes avec d'autres chaînes de requête. Pour plus d’informations, consultez ??? et ???.

Exemple de requête 1

Cet exemple suppose que les données de localisation sont stockées dans une shadow nommée `gps-tracker`. Le résultat de cette commande est la liste des périphériques situés à une distance radiale de 15,5 km du point central avec des coordonnées (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Exemple de requête 2

Cet exemple suppose que les données de localisation sont stockées dans un shadow classique. Le résultat de cette commande est la liste des périphériques situés à une distance radiale de 15,5 km du point central avec des coordonnées (47.6204, -122.3491).

```
aws iot search-index --query-string \
"shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Exemple de requête 3

Cet exemple suppose que les données de localisation sont stockées dans un shadow classique. Le résultat de cette commande est la liste des appareils qui ne sont pas connectés et situés en dehors de la distance radiale de 15,5 km du point central avec des coordonnées (47.6204, -122.3491).

```
aws iot search-index --query-string \
"connectivity.connected:false AND (NOT
 shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km)"
```

Didacticiel de démarrage

Ce didacticiel explique comment utiliser l'indexation de flotte pour indexer vos données de localisation. Pour simplifier, vous créez un objet pour représenter votre appareil et vous stockez les données de localisation dans une shadow nommée, vous mettez à jour la configuration d'indexation des objets pour l'indexation des emplacements et vous exécutez des exemples de géorequêtes pour rechercher des appareils dans une limite radiale.

Ce didacticiel vous prendra environ 15 minutes.

Dans cette rubrique :

- Prérequis
- Créez des objets et shadow
- Mettre à jour la configuration de l'indexation des objets
- Exécuter une géorequête

Prérequis

- Installez la dernière version de AWS CLI.
- Familiarisez-vous avec l'indexation des emplacements et les géorequêtes, la gestion de l'indexation des objets et la syntaxe des requêtes.

Créez des objets et shadow

Vous créez un objet pour représenter votre appareil et une shadow nommée pour stocker ses données de localisation (coordonnées 47.61564, -122.33584).

1. Exécutez la commande suivante pour créer votre objet qui représente votre vélo nommé Bike-1. Pour plus d'informations sur la façon de créer un objet en utilisant AWS CLI, voir create-thing from AWS CLIReference.

```
aws iot create-thing --thing-name "Bike-1" \
--attribute-payload '{"attributes": {"model":"OEM-2302-12", "battery":"35",
 "acqDate":"06/09/23"}}'
```

   Le résultat de cette commande peut ressembler à ce qui suit :

```
{
    "thingName": "Bike-1",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/Bike-1",
    "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df"
}
```

2. Exécutez la commande suivante pour créer une shadow nommée afin de stocker les données de localisation du Bike-1 (coordonnées 47.61564, -122.33584). Pour plus d'informations sur la façon de créer une ombre nommée à l'aide de AWS CLI, reportez-vous update-thing-shadowà la section AWS CLIRéférence.

```
aws iot-data update-thing-shadow \
--thing-name Bike-1 \
--shadow-name Bike1-shadow \
--cli-binary-format raw-in-base64-out \
--payload '{"state":{"reported":{"coordinates":{"lat": 47.6153, "lon": -122.3333}}}}'
 \
"output.txt" \
```

   Cette commande ne produit aucune sortie. Pour afficher l'ombre nommée que vous avez créée, vous pouvez exécuter la commande CLI list-named-shadows-for-thing.

```
aws iot-data list-named-shadows-for-thing --thing-name Bike-1
```

   Le résultat de cette commande peut ressembler à ce qui suit :

```
{
    "results": [
        "Bike1-shadow"
    ],
```

```
    "timestamp": 1699574309
}
```

Mettre à jour la configuration de l'indexation des objets

Pour indexer vos données de localisation, vous devez mettre à jour la configuration d'indexation de votre objet afin d'inclure les données de localisation. Dans la mesure où vos données de localisation sont stockées dans une shadow nommée dans ce didacticiel, définissez `thingIndexingMode` sur `REGISTRY` (au minimum), définissez `namedShadowIndexingMode` sur `ON` et ajoutez vos données de localisation à la configuration. Dans cet exemple, vous devez ajouter le nom de la shadow que vous avez nommée et le chemin des données de localisation de la shadow vers `filter`.

1. Exécutez la commande pour mettre à jour votre configuration d'indexation pour l'indexation des emplacements.

```
aws iot update-indexing-configuration --cli-input-json '{
"thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY",
"thingConnectivityIndexingMode": "OFF",
"deviceDefenderIndexingMode": "OFF",
"namedShadowIndexingMode": "ON",
"filter": {
    "namedShadowNames": ["Bike1-shadow"],
    "geoLocations":[{
        "name":"shadow.name.Bike1-shadow.reported.coordinates"
    }]
},
"customFields": [
{ "name":"attributes.battery",
"type":"Number"}] } }'
```

   La commande ne génère pas de sortie. Vous devrez peut-être attendre un moment jusqu'à ce que la mise à jour soit terminée. Pour vérifier l'état, exécutez la commande CLI describe-index. Si le message `indexStatus` indique :`ACTIVE`, la mise à jour de l'indexation de votre objet est terminée.

2. Exécutez la commande pour vérifier votre configuration d'indexation. Cette étape est facultative.

```
aws iot get-indexing-configuration
```

   Le résultat se présentera comme suit :

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY",
        "thingConnectivityIndexingMode": "OFF",
        "deviceDefenderIndexingMode": "OFF",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "shadow.name.*.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "registry.version",
                "type": "Number"
            },
            {
                "name": "registry.thingTypeName",
                "type": "String"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            },
            {
                "name": "shadow.name.*.version",
                "type": "Number"
            },
            {
                "name": "thingName",
                "type": "String"
            },
            {
                "name": "thingId",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "attributes.battery",
                "type": "Number"
            }
        ],
        "filter": {
```

```
            "namedShadowNames": [
                "Bike1-shadow"
            ],
            "geoLocations": [
                {
                    "name": "shadow.name.Bike1-shadow.reported.coordinates",
                    "order": "LatLon"
                }
            ]
        }
    },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

Exécuter une géorequête

Vous avez maintenant mis à jour la configuration d'indexation de vos objets pour inclure les données de localisation. Essayez de créer des géorequêtes et de les exécuter pour voir si vous pouvez obtenir les résultats de recherche souhaités. Une doit respecter la Syntaxe de requête. Vous trouverez des exemples de géorequêtes utiles dans ???.

Dans l'exemple de commande suivant, vous utilisez la géorequête `shadow.name.Bike1-shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km` pour rechercher des appareils situés à une distance radiale de 15,5 km du point central avec des coordonnées (47.6204, -122.3491).

```
aws iot search-index --query-string "shadow.name.Bike1-
shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

Étant donné que vous disposez d'un appareil situé aux coordonnées « lat »: 47.6153, « lon »: -122.3333, qui se situe à une distance de 15,5 km du point central, vous devriez pouvoir voir cet appareil (Bike-1) dans la sortie. Le résultat se présentera comme suit :

```
{
    "things": [
        {
            "thingName": "Bike-1",
            "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df",
```

```
            "attributes": {
                "acqDate": "06/09/23",
                "battery": "35",
                "model": "OEM-2302-12"
            },
            "shadow": "{\"reported\":{\"coordinates\":{\"lat\":47.6153,\"lon
\":-122.3333}},\"metadata\":{\"reported\":{\"coordinates\":{\"lat\":{\"timestamp
\":1699572906},\"lon\":{\"timestamp\":1699572906}}}},\"hasDelta\":false,\"version\":1}"
        }
    ]
}
```


Pour de plus amples informations, veuillez consulter ???.

Métriques de la flotte

Les métriques de flotte sont une fonctionnalité de l'indexation des flottes, un service géré qui vous permet d'indexer, de rechercher et d'agréger les données de vos appareils. AWS IoT Vous pouvez utiliser les indicateurs de parc pour surveiller l'état global des appareils de votre parc CloudWatchau fil du temps, notamment en examinant le taux de déconnexion des appareils de votre parc ou l'évolution moyenne du niveau de batterie sur une période donnée.

À l'aide des métriques de flotte, vous pouvez créer des requêtes d'agrégation dont les résultats sont transmis en continu CloudWatchsous forme de métriques pour analyser les tendances et créer des alarmes. Pour vos tâches de surveillance, vous pouvez spécifier les requêtes d'agrégation de différents types d'agrégation (statistiques, cardinalité et percentile). Vous pouvez enregistrer toutes vos requêtes d'agrégation afin de créer des métriques de flotte à réutiliser à l'avenir.

Didacticiel de démarrage

Dans ce didacticiel, vous allez créer une métrique de flotte pour surveiller les températures de vos capteurs afin de détecter d'éventuelles anomalies. Lors de la création de la métrique de flotte, vous définissez une requête d'agrégation qui détecte le nombre de capteurs dont les températures dépassent 80 degrés Fahrenheit. Vous spécifiez la requête à exécuter toutes les 60 secondes et les résultats de la requête sont envoyés CloudWatch, où vous pouvez voir le nombre de capteurs présentant des risques potentiels de température élevée et définir des alarmes. Pour suivre ce tutoriel, vous utilisez AWS CLI.

Dans ce didacticiel, vous allez découvrir comment :

- Configuration
- Créer des métriques de flotte
- Afficher les métriques dans CloudWatch
- Nettoyage des ressources

Ce didacticiel vous prendra environ 15 minutes.

Prérequis

- Installez la dernière version de AWS CLI
- Familiarisez-vous avec l'interrogation de données agrégées
- Familiarisez-vous avec l'utilisation CloudWatch des métriques Amazon

Configuration

Pour utiliser les métriques de flotte, activez l'indexation de la flotte. Pour activer l'indexation de la flotte pour vos objets ou groupes d'objets avec des sources de données spécifiées et des configurations associées, suivez les instructions des sections Gestion de l'indexation des objets et Gestion de l'indexation des groupes d'objets.

Pour configurer

1. Exécutez la commande suivante pour activer l'indexation de la flotte et spécifier les sources de données à partir desquelles effectuer la recherche.

```
aws iot update-indexing-configuration \
--thing-indexing-configuration
 "thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.temperature,type=Num
{name=attributes.rackId,type=String},
{name=attributes.stateNormal,type=Boolean}],thingConnectivityIndexingMode=STATUS" \
```

   L'exemple de commande CLI précédent permet à l'indexation de flotte de prendre en charge la recherche de données de registre, de données shadow et de l'état de connectivité des objets à l'aide de l'index `AWS_Things`.

   La modification de configuration peut prendre quelques minutes. Vérifiez que l'indexation de votre flotte est activée avant de créer des indicateurs de flotte.

Pour vérifier si l'indexation de votre flotte a été activée, exécutez la commande CLI suivante :

```
aws --region us-east-1 iot describe-index --index-name "AWS_Things"
```

Pour de plus amples informations, veuillez consulter Activer l'indexation d'objet.

2. Exécutez le script bash suivant pour créer dix objets et les décrire.

```
# Bash script. Type `bash` before running in other shells.

Temperatures=(70 71 72 73 74 75 47 97 98 99)
Racks=(Rack1 Rack1 Rack2 Rack2 Rack3 Rack4 Rack5 Rack6 Rack6 Rack6)
IsNormal=(true true true true true true false false false false)

for ((i=0; i < 10; i++))
do
  thing=$(aws iot create-thing --thing-name "TempSensor$i" --attribute-
payload attributes="{temperature=${Temperatures[@]:$i:1},rackId=${Racks[@]:
$i:1},stateNormal=${IsNormal[@]:$i:1}}")
  aws iot describe-thing --thing-name "TempSensor$i"
done
```

Ce script crée dix éléments pour représenter dix capteurs. Chaque élément possède les attributs de `temperaturerackId`, et `stateNormal` comme décrit dans le tableau suivant :

Attribut	Type de données	Description
`temperature`	Nombre	Valeur de température en degrés Fahrenheit
`rackId`	Chaîne	ID du rack de serveurs contenant des capteurs
`stateNormal`	Booléen	Si la valeur de température du capteur est normale ou non

La sortie de ce script contient dix fichiers JSON. Un des fichiers JSON ressemble au suivant :

```
{
    "version": 1,
    "thingName": "TempSensor0",
    "defaultClientId": "TempSensor0",
    "attributes": {
        "rackId": "Rack1",
        "stateNormal": "true",
        "temperature": "70"
    },
    "thingArn": "arn:aws:iot:region:account:thing/TempSensor0",
    "thingId": "example-thing-id"
}
```

Pour plus d'informations, consultez Créer un objet.

Créer des métriques de flotte

Crée une métrique de flotte

1. Exécutez la commande suivante pour créer une métrique de flotte nommée*high_temp_FM*. Vous créez la métrique du parc pour surveiller le nombre de capteurs dont les températures dépassent 80 degrés Fahrenheit. CloudWatch

```
aws iot create-fleet-metric --metric-name "high_temp_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature >80" --period 60 --aggregation-
field "attributes.temperature" --aggregation-type name=Statistics,values=count
```

--métrique-nom

Types de données : Chaîne. Le paramètre `--metric-name` spécifie le nom d'une métrique de flotte. Dans cet exemple, vous créez une métrique de flotte nommée high_temp_FM.

-- requête-chaîne

Types de données : Chaîne. Le paramètre `--query-string` spécifie la chaîne de requête. Dans cet exemple, la chaîne de requête signifie interroger tous les éléments dont les noms commencent par TempSensoret avec des températures supérieures à 80 degrés Fahrenheit. Pour plus d'informations, consultez Syntaxe de requête.

--période

Type de données – Entier. Le paramètre `--period` indique le temps nécessaire pour récupérer les données agrégées en secondes. Dans cet exemple, vous spécifiez que la métrique de flotte que vous créez récupère les données agrégées toutes les 60 secondes.

--agrégation-champ

Types de données : Chaîne. Le paramètre `--aggregation-field` indique l'attribut à évaluer. Dans cet exemple, l'attribut de température doit être évalué.

--agrégation-type

Le paramètre `--aggregation-type` spécifie le résumé statistique à afficher dans la métrique de flotte. Pour vos tâches de surveillance, vous pouvez personnaliser les propriétés de requête d'agrégation pour les différents types d'agrégation (statistiques, cardinalité et percentile). Dans cet exemple, vous spécifiez le nombre pour le type d'agrégation et les statistiques pour renvoyer le nombre d'appareils dont les attributs correspondent à la requête, en d'autres termes, pour renvoyer le nombre d'appareils dont le nom commence par TempSensoret dont les températures sont supérieures à 80 degrés Fahrenheit. Pour plus d'informations, consultez la section Interrogation des données agrégées.

La sortie de cette commande ressemble à ce qui suit :

```
{
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "metricName": "high_temp_FM"
}
```

> **Note**
>
> L'affichage des points de données peut prendre un certain temps CloudWatch.

Pour en savoir plus sur la création d'une métrique de flotte, consultez Gérer les métriques de flotte.

Si vous ne parvenez pas à créer une métrique de flotte, consultez la section Résolution des problèmes liés aux métriques de flotte.

2. (Facultatif) Exécutez la commande suivante pour décrire la métrique de votre flotte nommée high_temp_FM:

```
aws iot describe-fleet-metric --metric-name "high_temp_FM"
```

La sortie de cette commande ressemble à ce qui suit :

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625249775.834,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625249775.834,
    "metricName": "high_temp_FM"
}
```

Afficher les statistiques de la flotte dans CloudWatch

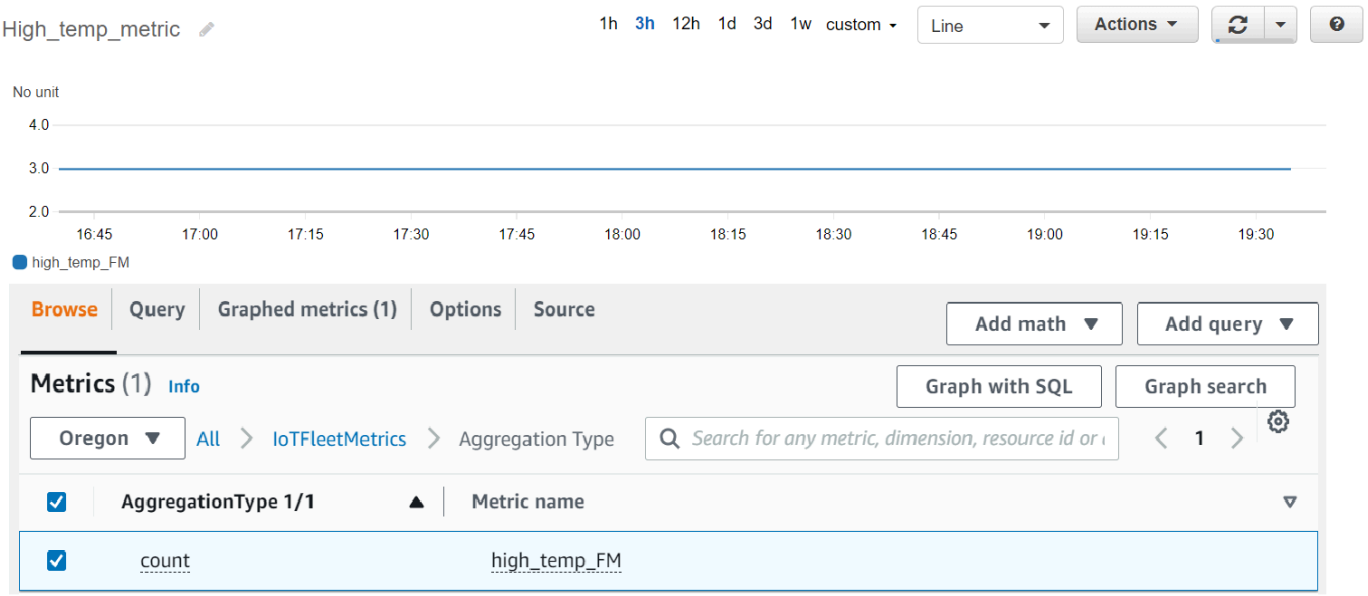
Après avoir créé la métrique de flotte, vous pouvez consulter les données de la métrique dans CloudWatch. Dans ce didacticiel, vous verrez la métrique qui indique le nombre de capteurs dont le nom commence par TempSensoret dont les températures sont supérieures à 80 degrés Fahrenheit.

Pour afficher les points de données dans CloudWatch

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. CloudWatch Dans le menu du panneau de gauche, choisissez Mesures pour développer le sous-menu, puis sélectionnez Toutes les mesures. Cela ouvre la page dont la moitié supérieure affiche le graphique et la moitié inférieure contient quatre sections à onglets.

3. La première section à onglets Toutes les métriques répertorie toutes les métriques que vous pouvez consulter par groupes, choisissez Io TFleet Metrics. Il contient tous les indicateurs de votre flotte.
4. Dans la section Type d'agrégation de l'onglet Toutes les métriques, choisissez Type d'agrégation pour afficher toutes les métriques de flotte que vous avez créées.
5. Choisissez la métrique de flotte pour afficher le graphique à gauche de la section Type d'agrégation. Vous verrez la valeur *count* à gauche du nom de votre métrique. Il s'agit de la valeur du type d'agrégation que vous avez spécifié dans la section Créer des métriques de flotte de ce didacticiel.
6. Choisissez le deuxième onglet intitulé Métriques graphiques à droite de l'onglet Toutes les métriques pour afficher la métrique de flotte que vous avez choisie à l'étape précédente.

   Vous devriez pouvoir voir un graphique qui affiche le nombre de capteurs dont la température est supérieure à 80 degrés Fahrenheit, comme suit :



> **Note**
>
> L'attribut Period est CloudWatch défini par défaut sur 5 minutes. Il s'agit de l'intervalle de temps entre les points de données affichés CloudWatch. Vous pouvez modifier le paramètre Période en fonction de vos besoins.

7. (Facultatif) Vous pouvez définir une alarme métrique.

1. CloudWatch Dans le menu du panneau de gauche, choisissez Alarmes pour développer le sous-menu, puis sélectionnez Toutes les alarmes.
2. Sur la page Alertes, choisissez Créer une alerte dans le coin supérieur droit. Suivez les instructions de création d'alerte dans la console pour créer une alerte selon vos besoins. Pour plus d'informations, consultez la section Utilisation des CloudWatch alarmes Amazon.

Pour en savoir plus, consultez l'article Utiliser CloudWatch les métriques Amazon.

Si vous ne pouvez pas voir les points de données CloudWatch, consultez la section Résolution des problèmes liés aux indicateurs de flotte.

Nettoyage

Pour supprimer les métriques de flotte

Vous utilisez la commande CLI delete-fleet-metric pour supprimer les métriques de flotte.

Pour supprimer la métrique de flotte nommée high_temp_FM, exécutez la commande suivante.

```
aws iot delete-fleet-metric --metric-name "high_temp_FM"
```

Pour nettoyer les objets

Vous pouvez utiliser la commande CLI delete-thing pour supprimer les objets.

Pour supprimer les dix éléments que vous avez créés, exécutez le script suivant :

```
# Bash script. Type `bash` before running in other shells.

for ((i=0; i < 10; i++))
do
  thing=$(aws iot delete-thing --thing-name "TempSensor$i")
done
```

Pour nettoyer les métriques dans CloudWatch

CloudWatch ne prend pas en charge la suppression des métriques. Les métriques expirent en fonction de leur calendrier de conservation. Pour en savoir plus, consultez la section Utilisation CloudWatch des métriques Amazon.

Gestion des métriques de flotte

Cette rubrique explique comment utiliser la AWS IoT console et comment AWS CLI gérer les indicateurs de votre flotte.

Rubriques

- Gestion des indicateurs de flotte (console)
- Gestion des indicateurs de flotte (CLI)
- Autoriser le balisage des ressources IoT

Gestion des indicateurs de flotte (console)

Les sections suivantes montrent comment utiliser la AWS IoT console pour gérer les indicateurs de votre flotte. Assurez-vous d'avoir activé l'indexation de la flotte avec les sources de données et les configurations associées avant de créer des métriques de flotte.

Activez l'indexation de la flotte

Si vous avez déjà activé l'indexation de la flotte, ignorez cette section.

Si vous n'avez pas activé l'indexation de la flotte, suivez ces instructions.

1. Ouvrez votre AWS IoT console à l'adresse https://console.aws.amazon.com/iot/.
2. AWS IoT Dans le menu, choisissez Réglages.
3. Pour afficher les paramètres détaillés, sur la page Paramètres, faites défiler la page vers le bas jusqu'à la section indexation de la flotte.
4. Pour mettre à jour les paramètres d'indexation de votre flotte, à droite de la section Indexation de la flotte, sélectionnez Gérer l'indexation.
5. Sur la page Gérer l'indexation de la flotte, mettez à jour les paramètres d'indexation de votre flotte en fonction de vos besoins.
   - Configuration

     Pour activer l'indexation des objets, activez l'indexation des objets, puis sélectionnez les sources de données à partir desquelles vous souhaitez indexer.

     Pour activer l'indexation des groupes d'objets, activez l'indexation des groupes d'objets.
   - Champs personnalisés pour l'agrégation - facultatif

Les champs personnalisés sont une liste de paires de noms et de types de champs.

Pour ajouter une paire de champs personnalisée, choisissez Ajouter un nouveau champ. Entrez un nom de champ personnalisé tel que`attributes.temperature`, puis sélectionnez un type de champ dans le menu Type de champ. Notez que le nom d'un champ personnalisé commence par `attributes.` et sera enregistré en tant qu'attribut pour exécuter des requêtes d'agrégation d'objets.

Pour mettre à jour et enregistrer le paramètre, choisissez Mettre à jour.

Créer une métrique de flotte

1. Ouvrez votre AWS IoT console à l'adresse https://console.aws.amazon.com/iot/.
2. AWS IoT Dans le menu, choisissez Gérer, puis choisissez Fleet metrics.
3. Sur la page Métriques de la flotte, choisissez Créer une métrique de flotte et suivre les étapes de création.
4. À l'étape 1, configurez les métriques de la flotte
   - Dans la section Requête, entrez une chaîne de requête pour spécifier les objets ou les groupes d'objets pour lesquels vous souhaitez effectuer la recherche agrégée. La chaîne de requête est composée d'un attribut et d'une valeur. Dans Propriétés, choisissez l'attribut souhaité ou, s'il n'apparaît pas dans la liste, saisissez-le dans le champ. Saisissez la valeur après `:`. Un exemple de chaîne de requête peut être `thingName:TempSensor*`. Pour chaque chaîne de requête que vous saisissez, appuyez sur la touche Entrée de votre clavier. Si vous entrez plusieurs chaînes de requête, spécifiez leur relation en sélectionnant et, ou, et non, ou pas entre elles.
   - Dans Propriétés du rapport, sélectionnez le nom de l'index, le type d'agrégation et le champ d'agrégation dans leurs listes respectives. Sélectionnez ensuite les données que vous souhaitez agréger dans Sélectionner les données, où vous pouvez sélectionner plusieurs valeurs de données.
   - Choisissez Suivant.
5. À l'étape 2, spécifiez les propriétés métriques de la flotte
   - Dans le champ Nom de la métrique de flotte, entrez le nom de la métrique de flotte que vous créez.
   - Dans le champ Description - facultatif, entrez une description de la métrique de flotte que vous créez. Ce champ est facultatif.

- Dans les champs Heures et Minutes, entrez l'heure (fréquence) à laquelle vous souhaitez que la métrique du parc émette des données CloudWatch.
- Choisissez Suivant.

6. À l'étape 3 : Examen et création
   - Vérifiez les paramètres des étapes 1 et 2. Pour modifier les paramètres, choisissez Modifier.
   - Choisissez Créer une métrique de flotte.

Une fois la création réussie, la métrique de flotte est répertoriée sur la page Métrique de flotte.

Met à jour une métrique de flotte

1. Sur la page Métrique de flotte, choisissez la métrique de flotte que vous souhaitez mettre à jour.
2. Sur la page métrique de la flotte, choisissez Détails, puis Modifier. Cela ouvre les étapes de création où vous pouvez mettre à jour les indicateurs de votre flotte dans l'une des trois étapes.
3. Une fois que vous avez terminé de mettre à jour la métrique de flotte, choisissez Mettre à jour la métrique de flotte.

Supprimer une métrique de flotte

1. Sur la page Métrique de flotte, choisissez la métrique de flotte que vous souhaitez mettre à jour.
2. Sur la page suivante qui affiche les détails de l'indicateur de votre flotte, choisissez Supprimer.
3. Dans la boîte de dialogue, entrez le nom de la métrique de flotte pour confirmer la suppression.
4. Sélectionnez Delete (Supprimer). Cette étape supprime définitivement la métrique de votre flotte.

Gestion des indicateurs de flotte (CLI)

Les sections suivantes montrent comment utiliser le AWS CLI pour gérer les indicateurs de votre flotte. Assurez-vous d'avoir activé l'indexation de la flotte avec les sources de données et les configurations associées avant de créer des métriques de flotte. Pour activer l'indexation de flotte pour vos objets ou groupes d'objets, suivez les instructions dans Gestion de l'indexation des objets ou Gestion de l'indexation des groupes d'objets.

Créer une métrique de flotte

Vous pouvez utiliser la commande create-fleet-metric CLI pour créer une métrique de flotte.

```
aws iot create-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 60 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum
```

La sortie de cette commande contient le nom et l'Amazon Resource Name (ARN) de la métrique de votre flotte. Le résultat se présente comme suit :

```
{
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "metricName": "YourFleetMetricName"
}
```

Liste des métriques de flotte

Vous pouvez utiliser la commande list-fleet-metric CLI pour répertorier tous les indicateurs de flotte de votre compte.

```
aws iot list-fleet-metrics
```

Le résultat de cette commande contient tous les indicateurs de votre flotte. Le résultat se présente comme suit :

```
{
    "fleetMetrics": [
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric1",
            "metricName": "YourFleetMetric1"
        },
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric2",
            "metricName": "YourFleetMetric2"
        }
    ]
}
```

Décrire une métrique de flotte

Vous pouvez utiliser la commande describe-fleet-metric CLI pour afficher des informations plus détaillées sur une métrique de flotte.

```
aws iot describe-fleet-metric --metric-name "YourFleetMetricName"
```

La sortie de la commande contient les informations détaillées sur la métrique de flotte spécifiée. Le résultat se présente comme suit :

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625790642.355,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "sum"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625790642.355,
    "metricName": "YourFleetMetricName"
}
```

Met à jour une métrique de flotte

Vous pouvez utiliser la commande update-fleet-metric CLI pour mettre à jour une métrique de flotte.

```
aws iot update-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 120 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum,count --index-name AWS_Things
```

La update-fleet-metric commande ne génère pas de sortie. Vous pouvez utiliser la commande describe-fleet-metric CLI pour voir le résultat.

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625792300.881,
    "queryString": "*",
    "period": 120,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
```

```
    "version": 2,
    "aggregationType": {
        "values": [
            "sum",
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625792300.881,
    "metricName": "YourFleetMetricName"
}
```

Supprimez une métrique de flotte

Utilisez la commande delete-fleet-metric CLI pour supprimer une métrique de flotte.

```
aws iot delete-fleet-metric --metric-name "YourFleetMetricName"
```

Cette commande ne produit aucune sortie si la suppression est réussie ou si vous spécifiez une métrique de flotte qui n'existe pas.

Pour plus d'informations, consultez Dépannage des métriques de flotte.

Autoriser le balisage des ressources IoT

Pour mieux contrôler les indicateurs de flotte que vous pouvez créer, modifier ou utiliser, vous pouvez associer des balises aux indicateurs de flotte.

Pour étiqueter les métriques de flotte que vous créez à l'aide de AWS Management Console ou AWS CLI, vous devez inclure l'`iot:TagResource`action dans votre politique IAM afin d'accorder des autorisations aux utilisateurs. Si votre politique IAM ne l'inclut pas `iot:TagResource`, toute action visant à créer une métrique de flotte avec une balise renverra une erreur `AccessDeniedException`.

Pour obtenir des informations générales sur le balisage de vos ressources, consultez Balisage de vos AWS IoT ressources.

Exemple de politique IAM

Reportez-vous à l'exemple de politique IAM suivant accordant des autorisations de balisage lorsque vous créez une métrique de flotte :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Action": [
    "iot:TagResource"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:fleetmetric/*"
   ]
  },
  {
   "Action": [
    "iot:CreateFleetMetric"
   ],
   "Effect": "Allow",
   "Resource": [
    "arn:aws:iot:*:*:index/*",
    "arn:aws:iot:*:*:fleetmetric/*"
   ]
  }
 ]
}
```

Pour de plus amples informations, consultez Actions, ressources et clés de condition pour AWS IoT

Livraison de fichiers basée sur MQTT

L'une des options que vous pouvez utiliser pour gérer les fichiers et les transférer vers les AWS IoT appareils de votre parc est la livraison de fichiers basée sur MQTT. Grâce à cette fonctionnalité dans le AWS cloud, vous pouvez créer un flux contenant plusieurs fichiers, mettre à jour les données du flux (liste et descriptions des fichiers), obtenir les données du flux, etc. AWS IoT La livraison de fichiers basée sur MQTT permet de transférer des données par petits blocs vers vos appareils IoT, en utilisant le protocole MQTT avec prise en charge des messages de demande et de réponse au format JSON ou CBOR.

Pour plus d'informations sur les moyens de transférer des données vers et depuis des appareils IoT à AWS IoT l'aide deConnectez des appareils à AWS IoT.

Rubriques

- Qu'est-ce qu'un stream ?
- Gestion d'un flux dans le AWS cloud
- Utilisation de AWS IoT la livraison de fichiers basée sur MQTT sur les appareils
- Exemple de cas d'utilisation dans FreeRTOS OTA

Qu'est-ce qu'un stream ?

Dans AWS IoT, un flux est une ressource accessible au public qui est une abstraction d'une liste de fichiers pouvant être transférés vers un appareil IoT. Un flux typique contient les informations suivantes :

- Un Amazon Resource Name (ARN) qui identifie de façon unique un flux à un moment donné. Cet ARN présente le modèle `arn:`*`partition`*`:iot:`*`region`*`:`*`account-ID`*`:stream/`*`stream ID`*.
- Un identifiant de flux qui identifie votre flux et qui est utilisé (et généralement requis) dans les commandes AWS Command Line Interface (AWS CLI) ou du SDK.
- Description du flux qui fournit une description de la ressource du flux.
- Une version de flux qui identifie une version particulière du flux. Comme les données du flux peuvent être modifiées immédiatement avant que les appareils ne commencent le transfert de données, la version du flux peut être utilisée par les appareils pour appliquer un contrôle de cohérence.

- Liste des fichiers pouvant être transférés vers des appareils. Pour chaque fichier de la liste, le flux enregistre un ID de fichier, la taille du fichier et les informations d'adresse du fichier, qui comprennent, par exemple, le nom du compartiment Amazon S3, la clé de l'objet et la version de l'objet.
- Rôle AWS Identity and Access Management (IAM) qui accorde à la livraison de fichiers AWS IoT basée sur MQTT l'autorisation de lire les fichiers de flux stockés dans le stockage de données.

AWS IoT La livraison de fichiers basée sur MQTT fournit les fonctionnalités suivantes afin que les appareils puissent transférer des données depuis le cloud : AWS

- Transfert de données à l'aide du protocole MQTT.
- Support des formats JSON ou CBOR.
- Possibilité de décrire un flux (`DescribeStream`API) pour obtenir la liste des fichiers de flux, la version du flux et les informations associées.
- Possibilité d'envoyer des données en petits blocs (`GetStream`API) afin que les appareils soumis à des contraintes matérielles puissent recevoir les blocs.
- Support d'une taille de bloc dynamique par demande, pour prendre en charge les appareils dotés de capacités de mémoire différentes.
- Optimisation pour les demandes de streaming simultanées lorsque plusieurs appareils demandent des blocs de données provenant du même fichier de flux.
- Amazon S3 comme stockage de données pour les fichiers de flux.
- Support pour la publication des journaux de transfert de données depuis la livraison de fichiers AWS IoT basée sur MQTT vers. CloudWatch

Pour les quotas de livraison de fichiers basés sur MQTT, consultez la section AWS IoT Core Service Quotas dans le Références générales AWS.

Gestion d'un flux dans le AWS cloud

AWS IoT fournit un AWS SDK et des AWS CLI commandes que vous pouvez utiliser pour gérer un flux dans le AWS cloud. Vous pouvez utiliser ces commandes pour effectuer les opérations suivantes :

- Créer un flux. CLI / SDK
- Décrivez un flux pour obtenir ses informations. CLI / SDK

- Répertoriez les flux dans votre Compte AWS. CLI / SDK
- Mettez à jour la liste des fichiers ou la description du flux dans un flux. CLI / SDK
- Supprime un flux. CLI / SDK

> **Note**
>
> À l'heure actuelle, les flux ne sont pas visibles dans le AWS Management Console. Vous devez utiliser le AWS SDK AWS CLI ou pour gérer un flux entrant. AWS IoT De plus, le Embedded C SDK est le seul SDK qui prend en charge les transferts de fichiers basés sur MQTT.

Avant d'utiliser la livraison de fichiers AWS IoT basée sur MQTT depuis vos appareils, vous devez vous assurer que les conditions suivantes sont remplies pour vos appareils, comme indiqué dans les sections suivantes :

- Une politique reflétant les autorisations correctes requises pour transmettre des données via MQTT.
- Votre appareil peut se connecter au AWS IoT Device Gateway.
- Déclaration de politique indiquant que vous pouvez étiqueter les ressources. S'il `CreateStream` est appelé avec des balises, `iot:TagResource` c'est obligatoire.

Avant d'utiliser la livraison de fichiers AWS IoT basée sur MQTT depuis vos appareils, vous devez suivre les étapes décrites dans les sections suivantes pour vous assurer que vos appareils sont correctement autorisés et peuvent se connecter à Device Gateway. AWS IoT

Accordez des autorisations à vos appareils

Vous pouvez suivre les étapes décrites dans Créer une AWS IoT politique pour créer une politique d'appareil ou utiliser une politique d'appareil existante. Attachez la stratégie aux certificats associés à vos appareils et ajoutez les autorisations suivantes à la stratégie d'appareil.

JSON

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

```
    "Statement": [
        {   "Effect": "Allow",
            "Action": [ "iot:Connect" ],
            "Resource": [
                "arn:partition:iot:region:accountID:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [ "iot:Receive", "iot:Publish" ],
            "Resource": [
                "arn:partition:iot:region:accountID:topic/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:partition:iot:region:accountID:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        }
    ]
}
```

Connectez vos appareils à AWS IoT

Les appareils qui utilisent la livraison de fichiers AWS IoT basée sur MQTT doivent se connecter à. AWS IoT AWS IoT La livraison de fichiers basée sur MQTT s'intègre AWS IoT au AWS cloud, de sorte que vos appareils doivent se connecter directement au point de terminaison du plan de données. AWS IoT

> **Note**
>
> Le point de terminaison du plan de AWS IoT données est spécifique à la région Compte AWS et. Vous devez utiliser le point de terminaison correspondant à la région dans laquelle vos appareils sont enregistrés AWS IoT. Compte AWS

Pour plus d'informations, consultez Connect à AWS IoT Core.

TagResource Usage

L'action `CreateStream` l'API crée un flux pour la livraison d'un ou de plusieurs fichiers volumineux par morceaux via MQTT.

Un appel `CreateStream` d'API réussi nécessite les autorisations suivantes :

- `iot:CreateStream`
- `iot:TagResource`(si `CreateStream` c'est avec des balises)

La politique qui prend en charge ces deux autorisations est indiquée ci-dessous :

JSON

```
{
 "Version": "2012-10-17",
 "Statement": {
     "Action": [ "iot:CreateStream", "iot:TagResource" ],
     "Effect": "Allow",
     "Resource": "arn:partition:iot:region:accountID:stream/streamId",
 }
}
```

L'action de la déclaration de `iot:TagResource` politique est requise pour garantir qu'un utilisateur ne puisse pas créer ou mettre à jour une balise sur une ressource sans les autorisations appropriées. Sans l'action de déclaration de politique spécifique à `iot:TagResource`, l'`CreateStream` appel d'API renverra un `AccessDeniedException` si la demande est accompagnée de balises.

Pour plus d'informations, consultez les liens suivantes :

- CreateStream
- TagResource
- Balise

Utilisation de AWS IoT la livraison de fichiers basée sur MQTT sur les appareils

Pour lancer le processus de transfert de données, un appareil doit recevoir un ensemble de données initial comprenant au minimum un identifiant de flux. Vous pouvez utiliser an AWS IoT Emplois pour planifier des tâches de transfert de données pour vos appareils en incluant l'ensemble de données initial dans le document de travail. Lorsqu'un appareil reçoit l'ensemble de données initial, il doit alors commencer à interagir avec la livraison de fichiers AWS IoT basée sur MQTT. Pour échanger des données avec la livraison de fichiers AWS IoT basée sur MQTT, un appareil doit :

- Utilisez le protocole MQTT pour vous abonner au Rubriques de livraison de fichiers basées sur MQTT.
- Envoyez des demandes, puis attendez de recevoir les réponses à l'aide de messages MQTT.

Vous pouvez éventuellement inclure un ID de fichier de flux et une version de flux dans l'ensemble de données initial. L'envoi d'un identifiant de fichier de flux à un appareil peut simplifier la programmation du microprogramme ou du logiciel de l'appareil, car il n'est plus nécessaire de faire une `DescribeStream` demande auprès de l'appareil pour obtenir cet identifiant. L'appareil peut spécifier la version du flux dans une `GetStream` demande afin d'appliquer un contrôle de cohérence au cas où le flux aurait été mis à jour de manière inattendue.

DescribeStream À utiliser pour obtenir des données de flux

AWS IoT La livraison de fichiers basée sur MQTT fournit l'`DescribeStream`API permettant d'envoyer des données de flux à un appareil. Les données de flux renvoyées par cette API incluent l'ID du flux, la version du flux, la description du flux et une liste de fichiers de flux, chacun ayant un ID de fichier et une taille de fichier en octets. Grâce à ces informations, un appareil peut sélectionner des fichiers arbitraires pour lancer le processus de transfert de données.

> **Note**
>
> Vous n'avez pas besoin d'utiliser l'`DescribeStream`API si votre appareil reçoit tous les fichiers de flux requis IDs dans l'ensemble de données initial.

Suivez ces étapes pour faire une `DescribeStream` requête.

1. Abonnez-vous au filtre de sujets « acceptés » `$aws/things/ThingName/streams/StreamId/description/json`.
2. Abonnez-vous au filtre de sujets « rejetés » `$aws/things/ThingName/streams/StreamId/rejected/json`.
3. Publiez une `DescribeStream` demande en envoyant un message à `$aws/things/ThingName/streams/StreamId/describe/json`.
4. Si la demande a été acceptée, votre appareil reçoit une `DescribeStream` réponse dans le filtre thématique « accepté ».
5. Si la demande a été rejetée, votre appareil reçoit une réponse d'erreur dans le filtre de rubrique « rejeté ».

> **Note**
>
> Si vous remplacez `json` par `cbor` dans les sujets et les filtres de sujets affichés, votre appareil reçoit les messages au format CBOR, qui est plus compact que le format JSON.

DescribeStream demande

Une `DescribeStream` demande au format JSON ressemble à l'exemple suivant.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039"
}
```

- (Facultatif) « `c` » est le champ du jeton client.

  La longueur du jeton client ne peut pas dépasser 64 octets. Un jeton client de plus de 64 octets provoque une réponse d'erreur et un message `InvalidRequest` d'erreur.

DescribeStream réponse

Une `DescribeStream` réponse dans JSON se présente comme suit.

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039",
    "s": 1,
```

```
    "d": "This is the description of stream ABC.",
    "r": [
        {
            "f": 0,
            "z": 131072
        },
        {
            "f": 1,
            "z": 51200
        }
    ]
}
```

- « c » est le champ du jeton client. Il est renvoyé s'il a été indiqué dans la `DescribeStream` demande. Utilisez le jeton client pour associer la réponse à sa demande.
- « s » est la version du flux sous forme d'entier. Vous pouvez utiliser cette version pour vérifier la cohérence de vos `GetStream` demandes.
- « r » contient la liste des fichiers du flux.
  - « f » est l'ID du fichier de flux sous forme d'entier.
  - « z » est la taille du fichier de flux en nombre d'octets.
- « d » Contient la description du flux.

Obtenir des blocs de données à partir d'un fichier de flux

Vous pouvez utiliser l'`GetStream`API pour qu'un appareil puisse recevoir des fichiers de flux sous forme de petits blocs de données, afin qu'elle puisse être utilisée par les appareils soumis à des contraintes de traitement de blocs de grande taille. Pour recevoir un fichier de données complet, un appareil peut avoir besoin d'envoyer ou de recevoir plusieurs demandes et réponses jusqu'à ce que tous les blocs de données soient reçus et traités.

GetStream demande

Suivez ces étapes pour faire une `GetStream` requête.

1. Abonnez-vous au filtre de sujets « acceptés » `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/data/json`.
2. Abonnez-vous au filtre de sujets « rejetés » `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/rejected/json`.

3. Publiez une `GetStream` demande dans le sujet `$aws/things/`*`ThingName`*`/streams/`*`StreamId`*`/get/json`.
4. Si la demande a été acceptée, votre appareil recevra une ou plusieurs `GetStream` réponses dans le filtre de sujets « accepté ». Chaque message de réponse contient des informations de base et une charge utile de données pour un seul bloc.
5. Répétez les étapes 3 et 4 pour recevoir tous les blocs de données. Vous devez répéter ces étapes si la quantité de données demandée est supérieure à 128 Ko. Vous devez programmer votre appareil pour qu'il utilise plusieurs `GetStream` requêtes afin de recevoir toutes les données demandées.
6. Si la demande a été rejetée, votre appareil recevra une réponse d'erreur dans le filtre de rubrique « rejeté ».

> **Note**
>
> - Si vous remplacez « json » par « cbor » dans les rubriques et les filtres de rubriques affichés, votre appareil recevra des messages au format CBOR, qui est plus compact que le format JSON.
> - AWS IoT La livraison de fichiers basée sur MQTT limite la taille d'un bloc à 128 Ko. Si vous faites une demande pour un bloc de plus de 128 Ko, la demande échouera.
> - Vous pouvez faire une demande pour plusieurs blocs dont la taille totale est supérieure à 128 Ko (par exemple, si vous faites une demande pour 5 blocs de 32 Ko chacun pour un total de 160 Ko de données). Dans ce cas, la demande n'échoue pas, mais votre appareil doit effectuer plusieurs demandes pour recevoir toutes les données demandées. Le service enverra des blocs supplémentaires au fur et à mesure que votre appareil fera des demandes supplémentaires. Nous vous recommandons de continuer avec une nouvelle demande uniquement une fois que la réponse précédente a été correctement reçue et traitée.
> - Quelle que soit la taille totale des données demandées, vous devez programmer votre appareil pour qu'il lance de nouvelles tentatives lorsque les blocs ne sont pas reçus ou ne sont pas reçus correctement.

Une `GetStream` demande au format JSON ressemble à l'exemple suivant.

```
{
```

```
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "s": 1,
    "f": 0,
    "l": 4096,
    "o": 2,
    "n": 100,
    "b": "..."
}
```

- [facultatif] « `c` » est le champ du jeton client.

  La longueur du jeton client ne peut pas dépasser 64 octets. Un jeton client de plus de 64 octets provoque une réponse d'erreur et un message `InvalidRequest` d'erreur.

- [facultatif] « `s` » est le champ de version du flux (un entier).

  La livraison de fichiers basée sur MQTT applique un contrôle de cohérence basé sur cette version demandée et sur la dernière version du flux dans le cloud. Si la version du flux envoyée depuis un appareil dans une `GetStream` demande ne correspond pas à la dernière version du flux dans le cloud, le service envoie une réponse d'erreur et un message `VersionMismatch` d'erreur. Généralement, un appareil reçoit la version de flux attendue (la plus récente) dans l'ensemble de données initial ou dans la réponse à `DescribeStream`.

- « `f` » est l'ID du fichier de flux (un entier compris entre 0 et 255).

  L'ID du fichier de flux est requis lorsque vous créez ou mettez à jour un flux à l'aide du SDK AWS CLI ou. Si un appareil demande un fichier de flux dont l'identifiant n'existe pas, le service envoie une réponse d'erreur et un message `ResourceNotFound` d'erreur.

- « `l` » est la taille du bloc de données en octets (un entier compris entre 256 et 131 072).

  Reportez-vous à Création d'un bitmap pour une requête GetStream pour obtenir des instructions sur l'utilisation des champs bitmap pour spécifier la partie du fichier de flux qui sera renvoyée dans la `GetStream` réponse. Si un appareil indique une taille de bloc hors de portée, le service envoie une réponse d'erreur et un message `BlockSizeOutOfBounds` d'erreur.

- [facultatif] « `o` » est le décalage du bloc dans le fichier de flux (un entier compris entre 0 et 98 304).

  Reportez-vous à Création d'un bitmap pour une requête GetStream pour obtenir des instructions sur l'utilisation des champs bitmap pour spécifier la partie du fichier de flux qui sera renvoyée dans la `GetStream` réponse. La valeur maximale de 98 304 est basée sur une limite de taille de fichier

de flux de 24 Mo et sur une taille de bloc minimale de 256 octets. La valeur par défaut est 0 si elle n'est pas spécifiée.

- [facultatif] « n » est le nombre de blocs demandés (un entier compris entre 0 et 98 304).

  Le champ « n » indique soit (1) le nombre de blocs demandés, soit (2) lorsque le champ bitmap (« b ») est utilisé, une limite du nombre de blocs qui seront renvoyés par la demande bitmap. Cette seconde utilisation est facultative. S'il n'est pas défini, sa valeur par défaut est *`DataBlockSize`* 131072/.

- [facultatif] « b » est un bitmap qui représente les blocs demandés.

  À l'aide d'un bitmap, votre appareil peut demander des blocs non consécutifs, ce qui facilite la gestion des nouvelles tentatives suite à une erreur. Reportez-vous à Création d'un bitmap pour une requête GetStream pour obtenir des instructions sur l'utilisation des champs bitmap pour spécifier la partie du fichier de flux qui sera renvoyée dans la `GetStream` réponse. Pour ce champ, convertissez le bitmap en une chaîne représentant la valeur du bitmap en notation hexadécimale. La bitmap doit être inférieure à 12 288 octets.

> **Important**
>
> Vous devez spécifier « b » ou « ». n Si aucune d'entre elles n'est spécifiée, la `GetStream` demande risque de ne pas être valide lorsque la taille du fichier est inférieure à 131072 octets (128 Ko).

GetStream réponse

Une `GetStream` réponse au format JSON ressemble à cet exemple pour chaque bloc de données demandé.

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "f": 0,
    "l": 4096,
    "i": 2,
    "p": "..."
}
```

- « `c` » est le champ du jeton client. Il est renvoyé s'il a été indiqué dans la `GetStream` demande. Utilisez le jeton client pour associer la réponse à sa demande.
- « `f` » est l'ID du fichier de flux auquel appartient la charge utile du bloc de données actuel.
- « `l` » est la taille de la charge utile du bloc de données en octets.
- « `i` » est l'ID du bloc de données contenu dans la charge utile. Les blocs de données sont numérotés à partir de 0.
- « `p` » contient la charge utile du bloc de données. Ce champ est une chaîne qui représente la valeur du bloc de données dans le codage Base64.

Création d'un bitmap pour une requête GetStream

Vous pouvez utiliser le champ bitmap (b) dans une `GetStream` demande pour obtenir des blocs non consécutifs à partir d'un fichier de flux. Cela permet aux appareils dont la capacité de RAM est limitée de faire face aux problèmes de distribution réseau. Un appareil ne peut demander que les blocs qui n'ont pas été reçus ou qui n'ont pas été reçus correctement. Le bitmap détermine les blocs du fichier de flux qui seront renvoyés. Pour chaque bit défini sur 1 dans le bitmap, un bloc correspondant du fichier de flux sera renvoyé.

Voici un exemple de spécification d'un bitmap et de ses champs d'appui dans une `GetStream` demande. Par exemple, vous souhaitez recevoir un fichier de flux en blocs de 256 octets (la taille du bloc). Imaginez que chaque bloc de 256 octets possède un numéro qui indique sa position dans le fichier, à partir de 0. Le bloc 0 est donc le premier bloc de 256 octets du fichier, le bloc 1 est le second, et ainsi de suite. Vous souhaitez demander les blocs 20, 21, 24 et 43 depuis le fichier.

Décalage de blocs

Le premier bloc étant le numéro 20, spécifiez le décalage (champo) sur 20 pour économiser de l'espace dans le bitmap.

Nombre total de verrous

Pour vous assurer que votre appareil ne reçoit pas plus de blocs qu'il ne peut en gérer avec des ressources de mémoire limitées, vous pouvez spécifier le nombre maximum de blocs qui doivent être renvoyés dans chaque message envoyé par livraison de fichiers basée sur MQTT. Notez que cette valeur n'est pas prise en compte si le bitmap lui-même indique un nombre inférieur à ce nombre de blocs, ou s'il est possible que la taille totale des messages de réponse envoyés par la distribution de fichiers basée sur MQTT soit supérieure à la limite de service de 128 Ko par `GetStream` requête.

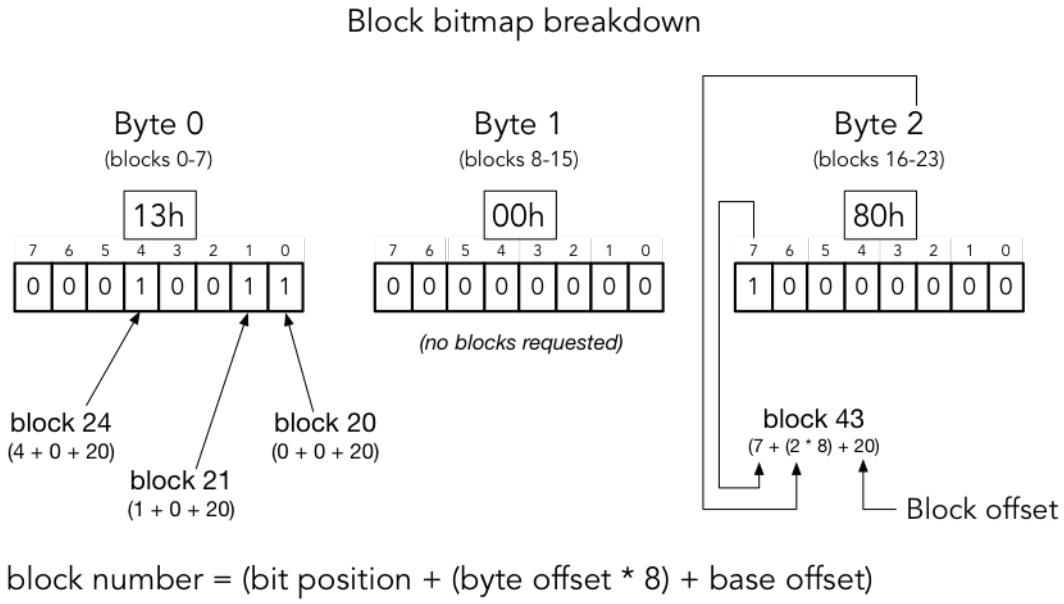
Bloquer le bitmap

Le bitmap lui-même est un tableau d'octets non signés exprimés en notation hexadécimale et inclus dans la `GetStream` demande sous forme de chaîne représentant le nombre. Mais pour construire cette chaîne, commençons par considérer le bitmap comme une longue séquence de bits (un nombre binaire). Si un bit de cette séquence est défini sur 1, le bloc correspondant du fichier de flux sera renvoyé à l'appareil. Dans notre exemple, nous voulons recevoir les blocs 20, 21, 24 et 43. Nous devons donc définir les bits 20, 21, 24 et 43 dans notre bitmap. Nous pouvons utiliser le décalage de bloc pour économiser de l'espace. Ainsi, après avoir soustrait le décalage de chaque numéro de bloc, nous voulons définir les bits 0, 1, 4 et 23, comme dans l'exemple suivant.

```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Si l'on prend un octet (8 bits) à la fois, cela s'écrit classiquement comme suit : « 0b00010011 », « 0b00000000 » et « 0b10000000 ». Le bit 0 apparaît dans notre représentation binaire à la fin du premier octet, et le bit 23 au début du dernier. Cela peut être source de confusion à moins que vous ne connaissiez les conventions. Le premier octet contient les bits 7-0 (dans cet ordre), le deuxième octet contient les bits 15-8, le troisième octet contient les bits 23-16, etc. En notation hexadécimale, cela se convertit en « 0x130080 ».

> **Tip**
>
> Vous pouvez convertir le binaire standard en notation hexadécimale. Prenez quatre chiffres binaires à la fois et convertissez-les en leur équivalent hexadécimal. Par exemple, « 0001 » devient « 1 », « 0011 » devient « 3 » et ainsi de suite.



En mettant tout cela ensemble, le JSON de notre `GetStream` requête ressemble à ce qui suit.

```
{
    "c" : "1",
    "s" : 1,
    "l" : 256,
    "f" : 1,
    "o" : 20,
    "n" : 32,
    "b" : "130080"
}
```

- « `c` » est le champ du jeton client.
- « `s` » est la version de diffusion attendue.
- « `l` » est la taille de la charge utile du bloc de données en octets.
- « `f` » est l'ID de l'index du fichier source.
- « o » est le décalage du bloc.
- « n » est le nombre de blocs.
- « b » est le bitmap BlockID manquant à partir du décalage. La valeur doit être codée en base64.

Gestion des erreurs liées à la livraison de AWS IoT fichiers basée sur MQTT

Réponse d'erreur envoyée à un appareil pour les deux `DescribeStream` et `GetStream` APIs contenant un jeton client, un code d'erreur et un message d'erreur. Une réponse d'erreur typique ressemble à l'exemple suivant.

```
{
    "o": "BlockSizeOutOfBounds",
    "m": "The block size is out of bounds",
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380"
}
```

- « o » est le code d'erreur qui indique la raison pour laquelle une erreur s'est produite. Reportez-vous aux codes d'erreur plus loin dans cette section pour plus de détails.
- « `m` » est le message d'erreur qui contient les détails de l'erreur.
- « `c` » est le champ du jeton client. Cela peut être retourné s'il a été indiqué dans la `DescribeStream` demande. Vous pouvez utiliser le jeton client pour associer la réponse à sa demande.

  Le champ du jeton client n'est pas toujours inclus dans une réponse d'erreur. Lorsque le jeton client indiqué dans la demande n'est pas valide ou est mal formé, il n'est pas renvoyé dans la réponse d'erreur.

> **Note**
>
> Pour des raisons de rétrocompatibilité, les champs de la réponse d'erreur peuvent être sous une forme non abrégée. Par exemple, le code d'erreur peut être désigné par les champs « code » ou « o » et le champ du jeton client peut être désigné par les champs « ClientToken » ou « c ». Nous vous recommandons d'utiliser la forme d'abréviation présentée ci-dessus.

InvalidTopic

La rubrique MQTT du message de diffusion n'est pas valide.

InvalidJson

La demande Stream n'est pas un document JSON valide.

InvalidCbor

La demande Stream n'est pas un document CBOR valide.

InvalidRequest

La demande est généralement identifiée comme étant mal formée. Pour plus d'informations, voir le message d'erreur.

Non autorisé

La demande n'est pas autorisée à accéder aux fichiers de données de flux sur le support de stockage, tel qu'Amazon S3. Pour plus d'informations, voir le message d'erreur.

BlockSizeOutOfBounds

La taille du bloc est hors limites. Reportez-vous à la section « Livraison de fichiers basée sur MQTT » dans AWS IoT Core Service Quotas.

OffsetOutOfBounds

Le décalage est hors limites. Reportez-vous à la section « Livraison de fichiers basée sur MQTT » dans AWS IoT Core Service Quotas.

BlockCountLimitExceeded

Le nombre de blocs de requêtes est hors limites. Reportez-vous à la section « Livraison de fichiers basée sur MQTT » dans AWS IoT Core Service Quotas.

BlockBitmapLimitExceeded

La taille du bitmap de demande est hors limites. Reportez-vous à la section « Livraison de fichiers basée sur MQTT » dans AWS IoT Core Service Quotas.

ResourceNotFound

Le flux, les fichiers, les versions de fichiers ou les blocs demandés sont introuvables. Reportez-vous au message d'erreur pour plus de détails.

VersionMismatch

La version du flux dans la demande ne correspond pas à la version du flux dans la fonctionnalité de livraison de fichiers basée sur MQTT. Cela indique que les données du flux ont été modifiées depuis que la version du flux a été initialement reçue par l'appareil.

ETagIncompatibilité

Le S3 ETag du flux ne correspond pas à celui ETag de la dernière version de l'objet S3.

InternalError

Une erreur interne s'est produite lors de la livraison de fichiers basée sur MQTT.

Exemple de cas d'utilisation dans FreeRTOS OTA

L'agent FreeRTOS OTA over-the-air () AWS IoT utilise la livraison de fichiers basée sur MQTT pour transférer les images du microprogramme FreeRTOS vers des appareils FreeRTOS. Pour envoyer l'ensemble de données initial à un appareil, celui-ci utilise le service AWS IoT Job pour planifier une tâche de mise à jour OTA sur les appareils FreeRTOS.

Pour une implémentation de référence d'un client de distribution de fichiers basé sur MQTT, consultez les codes d'agent FreeRTOS OTA dans la documentation FreeRTOS.

Device Advisor

Device Advisor est une fonctionnalité de test entièrement gérée basée sur le cloud qui permet de valider les appareils IoT lors du développement du logiciel des appareils. Device Advisor propose des tests prédéfinis que vous pouvez utiliser pour valider les appareils IoT afin de garantir une connectivité fiable et sécurisée AWS IoT Core, avant de les déployer en production. Les tests prédéfinis de Device Advisor vous aident à valider le logiciel de votre appareil par rapport aux meilleures pratiques en matière d'utilisation des protocoles TLS, MQTT, Device Shadow, et IoT Jobs. Vous pouvez également télécharger des rapports de qualification signés à soumettre au AWS Partner Network afin que votre appareil soit qualifié pour le AWS Partner Device Catalog (catalogue d'appareils partenaires) sans avoir à envoyer votre appareil et à attendre qu'il soit testé.

> **Note**
>
> Device Advisor est pris en charge dans les régions us-west-1 et eu-west-1.
> Device Advisor prend en charge les appareils et les clients qui utilisent les protocoles MQTT et MQTT over WebSocket Secure (WSS) pour publier des messages et s'y abonner. Tous les protocoles prennent en charge IPv4 et IPv6.
> Device Advisor prend en charge les certificats de serveur RSA.

Tout appareil conçu pour se connecter AWS IoT Core peut tirer parti de Device Advisor. Vous pouvez accéder à Device Advisor depuis la AWS IoT console ou à l'aide du AWS CLI SDK. Lorsque vous êtes prêt à tester votre appareil, enregistrez-le auprès du terminal Device Advisor AWS IoT Core et configurez-le avec le logiciel de l'appareil. Choisissez ensuite les tests prédéfinis, configurez-les, exécutez les tests sur votre appareil et obtenez les résultats des tests ainsi que des journaux détaillés ou un rapport de qualification.

Device Advisor est un point de terminaison de test dans le AWS cloud. Vous pouvez tester vos appareils en les configurant pour qu'ils se connectent au point de terminaison de test fourni par le Device Advisor. Une fois qu'un appareil est configuré pour se connecter au point de terminaison de test, vous pouvez accéder à la console Device Advisor ou utiliser le AWS SDK pour choisir les tests que vous souhaitez exécuter sur vos appareils. Device Advisor gère ensuite le cycle de vie complet d'un test, y compris le provisionnement des ressources, la planification du processus de test, la gestion de la machine d'état, l'enregistrement du comportement de l'appareil, l'enregistrement des résultats et la fourniture des résultats finaux sous forme de rapport de test.

Protocoles TLS

Le protocole TLS (Transport Layer Security) est utilisé pour chiffrer les données confidentielles sur des réseaux non sécurisés comme Internet. Le protocole TLS est le successeur du protocole Secure Sockets Layer (SSL).

Device Advisor prend en charge les protocoles TLS suivants :

- TLS 1.3 (recommandé)
- TLS 1.2

Protocols, port mappings, and authentication (Protocoles, mappages de ports et authentification)

Le protocole de communication de l'appareil est utilisé par un appareil ou un client pour se connecter au courtier de messages en utilisant un point de terminaison du dispositif. Le tableau suivant répertorie les protocoles pris en charge par les points de terminaison Device Advisor ainsi que les méthodes d'authentification et les ports utilisés.

Protocoles, authentification et mappages de port

Protocole	Opérations prises en charge	Authentification	Port	Nom du protocole ALPN
MQTT terminé WebSocket	Publier, S'abonner	Signature Version 4	443	N/A
MQTT	Publier, S'abonner	Certificat de client X.509	8883	`x-amzn-mqtt-ca`
MQTT	Publier, S'abonner	Certificat de client X.509	443	N/A

Ce chapitre contient les sections suivantes :

- Configuration
- Premiers pas avec Device Advisor dans la console
- Flux de travail Device Advisor
- Flux de travail détaillé sur la console Device Advisor
- Flux de travail de la console de tests de longue durée

- Points de terminaison d’un VPC Device Advisor (AWS PrivateLink)
- Cas de test Device Advisor

Configuration

Avant d'utiliser Device Advisor pour la première fois, effectuez les tâches suivantes :

Création d’un objet IoT

Tout d'abord, créez un objet IoT et associez un certificat. Pour un didacticiel sur la création d'objets, voir Création d'un d'objet.

Créer un rôle IAM à utiliser comme rôle de périphérique

> **Note**
>
> Vous pouvez rapidement créer le rôle d'appareil à l'aide de la console Device Advisor. Pour savoir comment configurer le rôle de votre appareil avec la console Device Advisor, consultez Commencer à utiliser Device Advisor dans la console.

1. Accédez à la AWS Identity and Access Management console et connectez-vous à celle que Compte AWS vous utilisez pour tester Device Advisor.
2. Dans le panneau de navigation de gauche, choisissez Politiques.
3. Choisissez Create Policy (Créer une politique).
4. Sous Review Policy (Examiner une politique), procédez comme suit :

    a. Pour Service, choisissez IoT.

    b. Sous Actions, effectuez l'une des opérations suivantes :

    - (Recommandé) Sélectionnez les actions en fonction de la politique associée à l'objet ou au certificat IoT que vous avez créé dans la section précédente.
    - Recherchez les actions suivantes dans Filter action (Action de filtrage) et sélectionnez-les :

        - `Connect`
        - `Publish`

- `Subscribe`
- `Receive`
- `RetainPublish`

c. Sous Ressources, limitez le client, le sujet et les ressources du sujet. Restreindre ces ressources constitue une bonne pratique de sécurité. Pour limiter les ressources, procédez comme suit :

i. Choisissez Spécifier l'ARN de la ressource client pour l'action Connect.

ii. Choisissez Ajouter un ARN, puis effectuez l'une des opérations suivantes :

> **Note**
>
> Le clientId est l'ID du client MQTT que votre appareil utilise pour interagir avec Device Advisor.

- Spécifiez la Région, accountID, et le clientID dans l'éditeur visuel ARN.
- Entrez manuellement les Amazon Resource Names (ARNs) des sujets IoT avec lesquels vous souhaitez exécuter vos scénarios de test.

iii. Choisissez Ajouter.

iv. Choisissez Specify topic resource ARN for the Receive and one more action. (Spécifiez l'ARN de la ressource de sujet pour la réception et une autre action)

v. Choisissez Ajouter un ARN, puis effectuez l'une des opérations suivantes :

> **Note**
>
> Le topic name (nom de rubrique) est lz rubrique MQTT sur lequel votre appareil publie des messages.

- Spécifiez la Région, accountID, et le Topic name dans l'éditeur visuel ARN.
- Entrez manuellement ARNs les sujets IoT avec lesquels vous souhaitez exécuter vos scénarios de test.

vi. Choisissez Ajouter.

vii. Choisissez Spécifier l'ARN de la ressource TopicFilter pour l'action Subscribe.

viii. Choisissez Ajouter un ARN, puis effectuez l'une des opérations suivantes :

> **Note**
>
> Le topic name (nom de rubrique) est la rubrique MQTT à laquelle votre appareil est abonné.

- Spécifiez la Région, accountID, et le Topic name dans l'éditeur visuel ARN.
- Entrez manuellement ARNs les sujets IoT avec lesquels vous souhaitez exécuter vos scénarios de test.

ix. Choisissez Ajouter.

5. Choisissez Suivant : Balises.
6. Choisissez Suivant : Vérification.
7. Sous Review policy, (Examiner une politique), saisissez un Nom pour votre politique.
8. Choisissez Create Policy (Créer une politique).
9. Dans le panneau de navigation de gauche, choisissez Rôles.
10. Choisissez Create Role (Créer un rôle).
11. Sous Select trusted entity (Sélectionner une entité de confiance) , choisissez Custom trust policy. (Stratégie de confiance personnalisée)
12. Entrez la politique de confiance suivante dans le champ Custom trust policy (Politique de confiance personnalisée) Ajoutez les clés de condition globale `aws:SourceArn` et `aws:SourceAccount` à la politique pour vous protéger contre le problème de l'adjoint confus.

> **Important**
>
> Votre `aws:SourceArn` doit se conformer à la section `format: arn:aws:iotdeviceadvisor:`*`region:account-id`*`:*`. Assurez-vous que *`region`* correspond à votre AWS IoT région et que *`account-id`* correspond à votre numéro de compte client. Pour plus d'informations, consultez Prévention du problème de l'adjoint confus entre services.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAwsIoTCoreDeviceAdvisor",
            "Effect": "Allow",
            "Principal": {
                "Service": "iotdeviceadvisor.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn":
 "arn:aws:iotdeviceadvisor:*:111122223333:suitedefinition/*"
                }
            }
        }
    ]
}
```

13. Choisissez Suivant.
14. Choisissez la politique que vous avez créée à l'étape 4.
15. (Facultatif) Sous Définir la limite d'autorisations, choisissez Utiliser une limite des autorisations pour contrôler les autorisations de rôle maximales, puis sélectionnez la stratégie que vous avez créée.
16. Choisissez Suivant.
17. Entrez un nom de rôle et une description.
18. Choisissez Créer un rôle.

Créez une politique gérée par le client pour qu'un utilisateur IAM puisse utiliser Device Advisor

1. Accédez à la console IAM à https://console.aws.amazon.com/iam/l'adresse. Si vous y êtes invité, saisissez vos informations d'identification AWS .
2. Dans le volet de navigation de gauche, choisissez Politiques.
3. Choisissez Create policy (Créer une politique), puis choisissez l'onglet JSON.
4. Ajoutez les autorisations nécessaires pour utiliser Device Advisor. Le document de politique se trouve dans la rubrique Bonnes pratiques en matière de sécurité.
5. Choisissez Review Policy (Examiner une stratégie).
6. Saisissez un Name (Nom) et une Description.
7. Choisissez Create Policy (Créer une stratégie).

Création d'un utilisateur IAM pour utiliser Device Advisor

> **Note**
>
> Nous vous recommandons de créer un utilisateur IAM à utiliser lorsque vous exécutez des tests Device Advisor. L'exécution de tests Device Advisor par un utilisateur administrateur peut présenter des risques de sécurité et n'est donc pas recommandée.

1. Accédez à la console IAM sur https://console.aws.amazon.com/iam/Si vous y êtes invité, entrez vos AWS informations d'identification pour vous connecter.
2. Dans le volet de navigation de gauche, choisissez Utilisateurs.
3. Sélectionnez Ajouter un utilisateur.
4. Entrez un nom d'utilisateur.
5. Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

   Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identification à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

6. Choisissez Suivant : Autorisations.
7. Pour activer l'accès, ajoutez des autorisations à vos utilisateurs, groupes ou rôles :

   - Utilisateurs et groupes dans AWS IAM Identity Center :

     Créez un jeu d'autorisations. Suivez les instructions de la rubrique Création d'un jeu d'autorisations du Guide de l'utilisateur AWS IAM Identity Center .

   - Utilisateurs gérés dans IAM par un fournisseur d'identité :

     Créez un rôle pour la fédération d'identité. Suivez les instructions de la rubrique Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM.

- Utilisateurs IAM :
  - Créez un rôle que votre utilisateur peut assumer. Suivez les instructions de la rubrique Création d'un rôle pour un utilisateur IAM dans le Guide de l'utilisateur IAM.
  - (Non recommandé) Attachez une politique directement à un utilisateur ou ajoutez un utilisateur à un groupe d'utilisateurs. Suivez les instructions de la rubrique Ajout d'autorisations à un utilisateur (console) du Guide de l'utilisateur IAM.

8. Entrez le nom de la politique gérée par le client que vous avez créée dans la zone de recherche. Cochez ensuite la case Nom de la stratégie.
9. Choisissez Suivant : Balises.
10. Choisissez Next: Review (Suivant : Vérification).
11. Choisissez Create user.
12. Choisissez Close (Fermer).

Device Advisor a besoin d'accéder à vos AWS ressources (objets, certificats et terminaux) en votre nom. Votre utilisateur IAM doit disposer des autorisations nécessaires. Device Advisor publiera également des journaux sur Amazon CloudWatch si vous associez la politique d'autorisation nécessaire à votre utilisateur IAM.

Configurer votre appareil

Device Advisor utilise l'extension TLS d'indication de nom de serveur (SNI) pour appliquer les configurations TLS. Les appareils doivent utiliser cette extension lorsqu'ils se connectent et transmettent un nom de serveur identique à celui du point de terminaison de test Device Advisor.

Device Advisor autorise la connexion TLS lorsqu'un test est en cours `Running`. Il refuse la connexion TLS avant et après chaque test. C'est pourquoi nous vous recommandons d'utiliser le mécanisme de nouvelle tentative de connexion de l'appareil pour une expérience de test entièrement automatisée avec Device Advisor. Vous pouvez exécuter des suites de tests qui incluent plusieurs scénarios de test, telles que TLS connect, MQTT connect et MQTT publish. Si vous exécutez plusieurs scénarios de test, nous recommandons que votre appareil essaie de se connecter à notre point de terminaison de test toutes les cinq secondes. Vous pouvez ensuite automatiser l'exécution de plusieurs scénarios de test en séquence.

> **Note**
>
> Pour préparer le logiciel de votre appareil à des fins de test, nous vous recommandons d'utiliser un SDK auquel vous pouvez vous connecter AWS IoT Core. Vous devez ensuite mettre à jour le SDK avec le point de terminaison de test Device Advisor fourni pour votre Compte AWS.

Device Advisor prend en charge deux types de points de terminaison : les points de terminaison au niveau du compte et au niveau de l'appareil. Choisissez le point de terminaison qui correspond le mieux à votre cas d'utilisation. Pour exécuter simultanément plusieurs suites de tests pour différents appareils, utilisez un point de terminaison au niveau de l'appareil.

Exécutez la commande suivante pour obtenir le point de terminaison au niveau de l'appareil :

Pour les clients MQTT utilisant des certificats client X.509 :

```
aws iotdeviceadvisor get-endpoint --thing-arn your-thing-arn
```

or

```
aws iotdeviceadvisor get-endpoint --certificate-arn your-certificate-arn
```

Pour les WebSocket clients MQTT over utilisant Signature Version 4 :

```
aws iotdeviceadvisor get-endpoint --device-role-arn your-device-role-arn --
authentication-method SignatureVersion4
```

Pour exécuter une suite de tests à la fois, choisissez un point de terminaison au niveau du compte. Exécutez la commande suivante pour obtenir le point de terminaison au niveau du compte :

```
aws iotdeviceadvisor get-endpoint
```

Premiers pas avec Device Advisor dans la console

Ce didacticiel vous aide à démarrer AWS IoT Core Device Advisor sur la console. Device Advisor propose des fonctionnalités telles que les tests obligatoires et les rapports de qualification signés. Vous pouvez utiliser ces tests et rapports pour qualifier et répertorier les appareils dans AWS Partner

Device Catalog (catalogue des appareils partenaires), comme indiqué dans le programme de AWS IoT Core qualification.

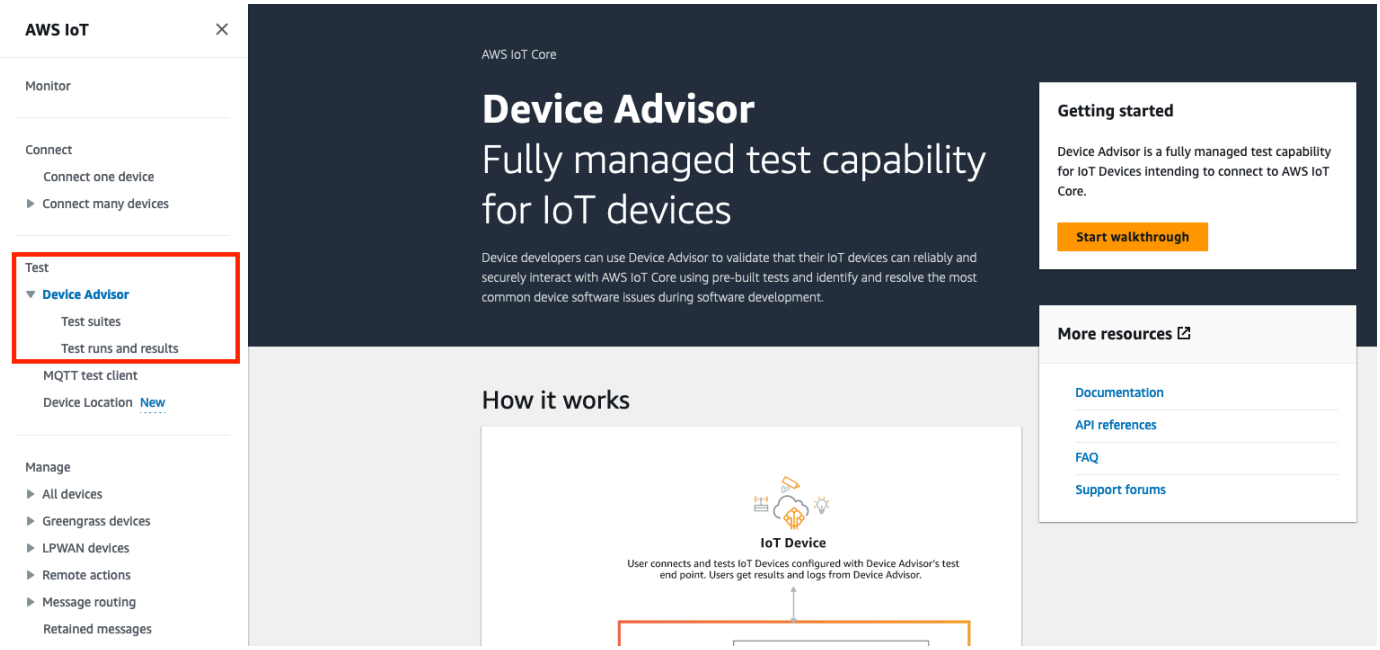
Pour plus d'informations sur l'utilisation de Device Advisor, consultez Flux de travail Device Advisor et Flux de travail détaillé sur la console Device Advisor.

Pour terminer ce didacticiel, suivez les étapes décrites dans Configuration.

> **Note**
>
> Device Advisor est compatible dans les domaines suivants Régions AWS :
>
> - USA Est (Virginie du Nord)
> - USA Ouest (Oregon)
> - Asie Pacifique (Tokyo)
> - Europe (Irlande)

Premiers pas

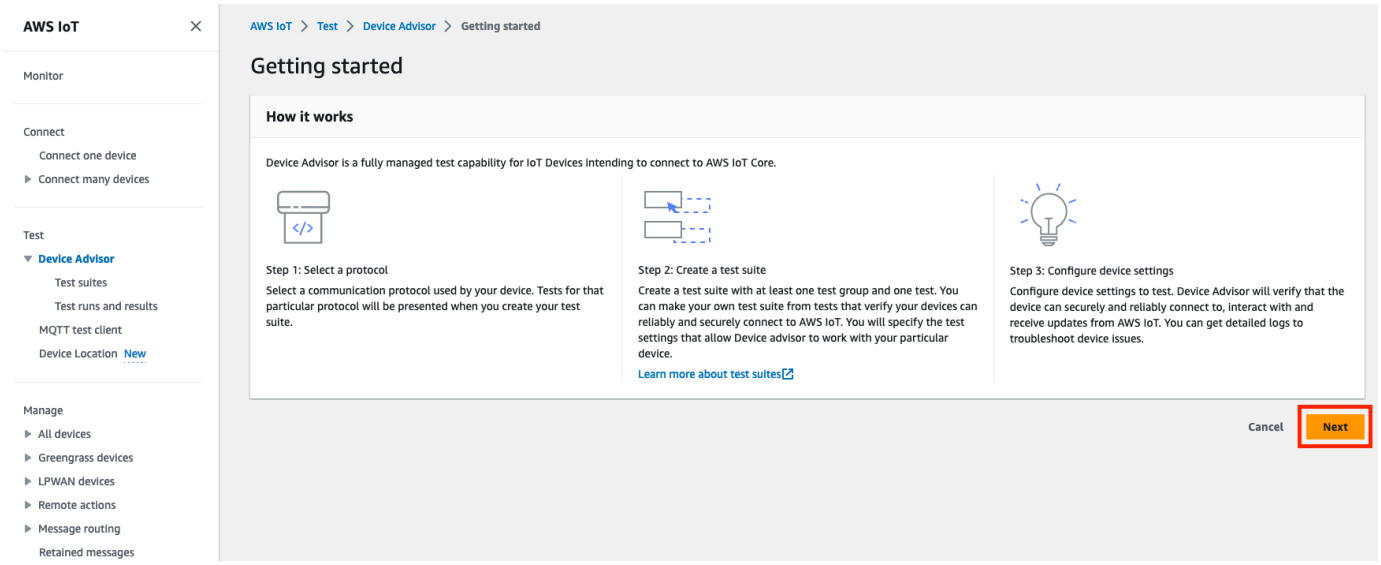
1. Dans le volet de navigation de laAWS IoT console, sous Test, choisissez Device Advisor. Cliquez ensuite sur le bouton Démarrer la procédure pas à pas sur la console.



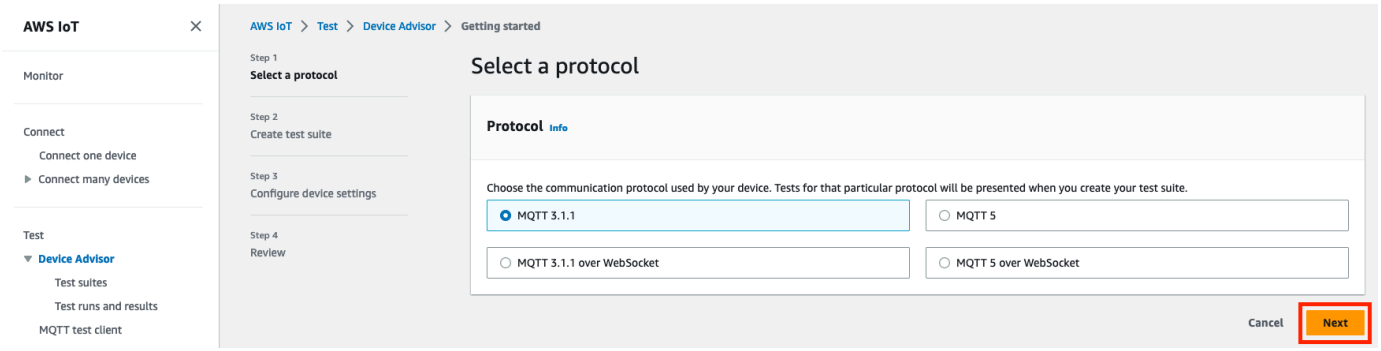
2. La page Commencer à utiliser Device Advisor fournit un aperçu des étapes nécessaires pour créer une suite de tests et exécuter des tests sur votre appareil. Vous trouverez également le

point de terminaison de test Device Advisor pour votre compte ici. Vous devez configurer le microprogramme ou le logiciel de l'appareil utilisé pour les tests afin de vous connecter à ce point de terminaison de test.

Pour terminer ce didacticiel, créez d'abord un objet et un certificat. Après avoir examiné les informations de la section Fonctionnement, choisissez Suivant.

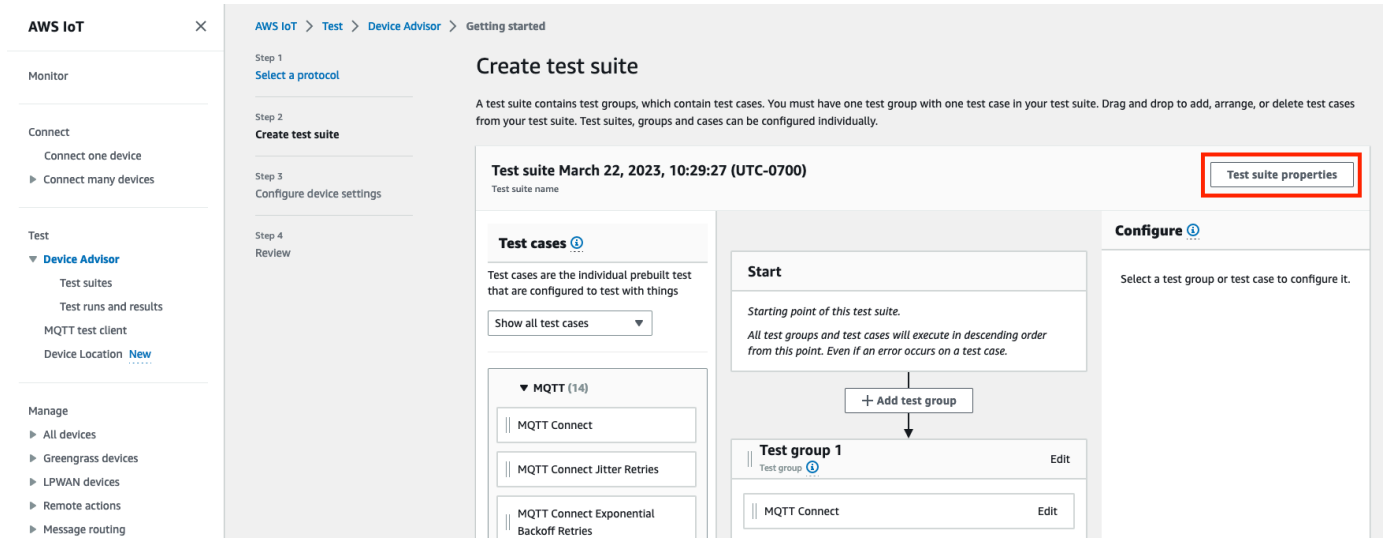


3. À l'étape 1 : Sélectionnez un protocole, sélectionnez un protocole parmi les options répertoriées. Ensuite, choisissez Suivant.



4. À l'étape 2, vous créez et configurez une suite de tests personnalisée. Une suite de tests personnalisée doit comporter au moins un groupe de test, et chaque groupe de test doit comporter au moins un cas de test. Nous avons ajouté le scénario de test MQTT Connect pour que vous puissiez commencer.

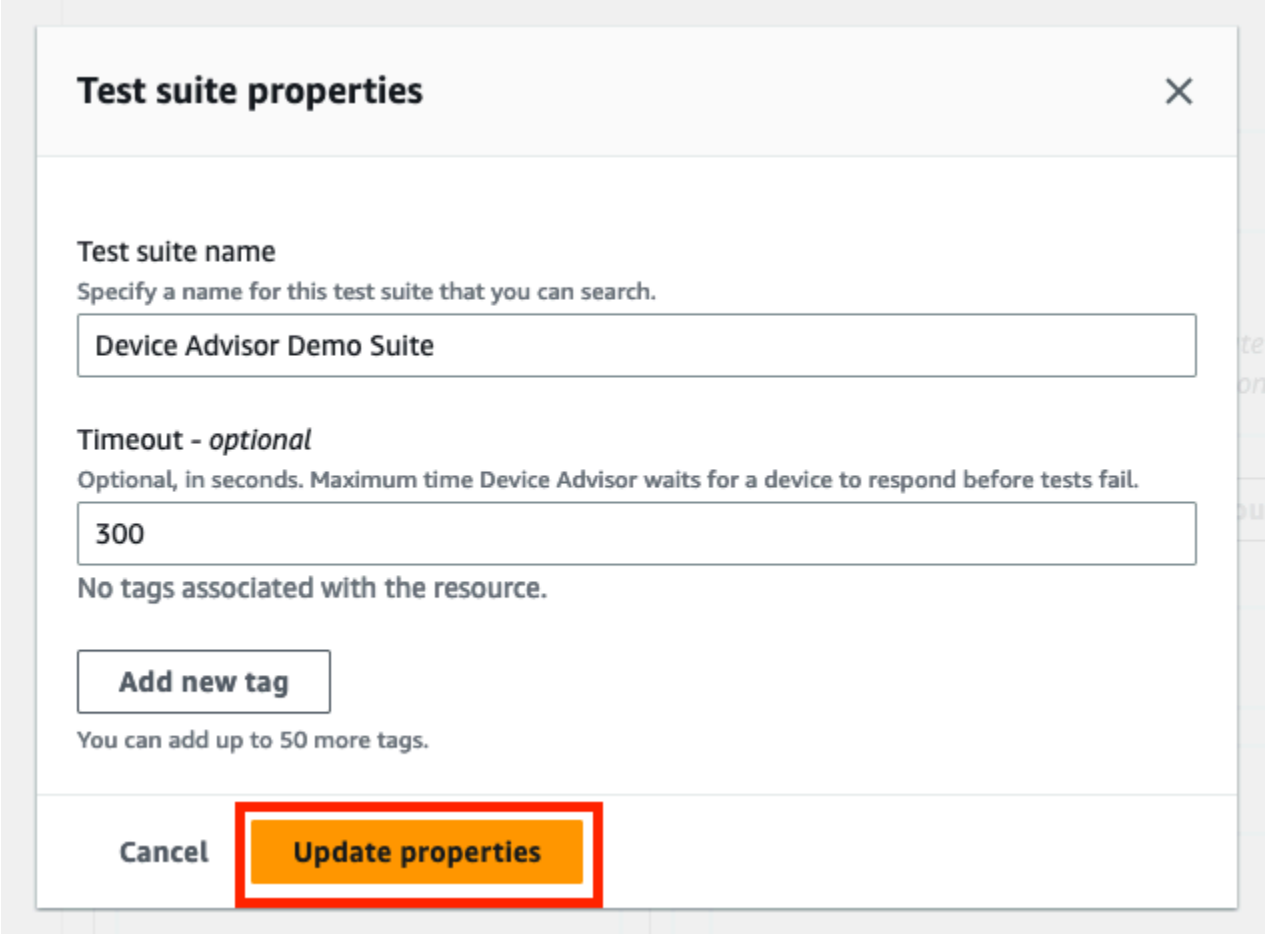
   Choisissez Propriétés de la suite de tests.



Indiquez les propriétés de la suite de tests lorsque vous créez votre suite de tests. Vous pouvez configurer les propriétés suivantes au niveau de la suite :

- Test suite name: (Nom de la suite de tests) entrez le nom de votre suite de tests.
- Timeout (optional) Délai(facultatif) délai d'expiration (en secondes) pour chaque scénario de test de la suite de tests actuelle. Si vous ne spécifiez pas de valeur de délai d'attente, la valeur par défaut est utilisée.
- Balises (facultatif) : ajoutez des balises à la suite de tests.

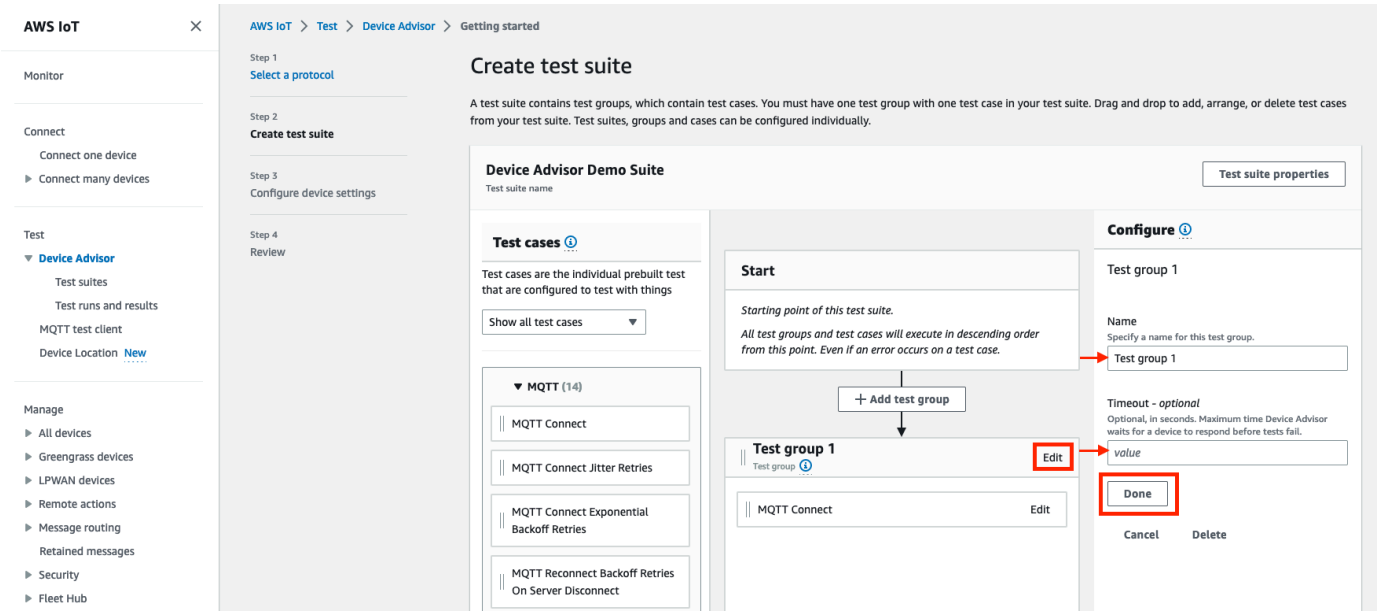
Lorsque vous avez terminé, choisissez Mettre à jour les propriétés.



5. (Facultatif) Pour mettre à jour la configuration du groupe de suites de tests, cliquez sur le bouton Modifier à côté du nom du groupe de tests.

   - Name (Nom) : Entrez un nom personnalisé pour le groupe de suites de tests.
   - Timeout (optional) Délai(facultatif) délai d'expiration (en secondes) pour chaque scénario de test de la suite de tests actuelle. Si vous ne spécifiez pas de valeur de délai d'attente, la valeur par défaut est utilisée.

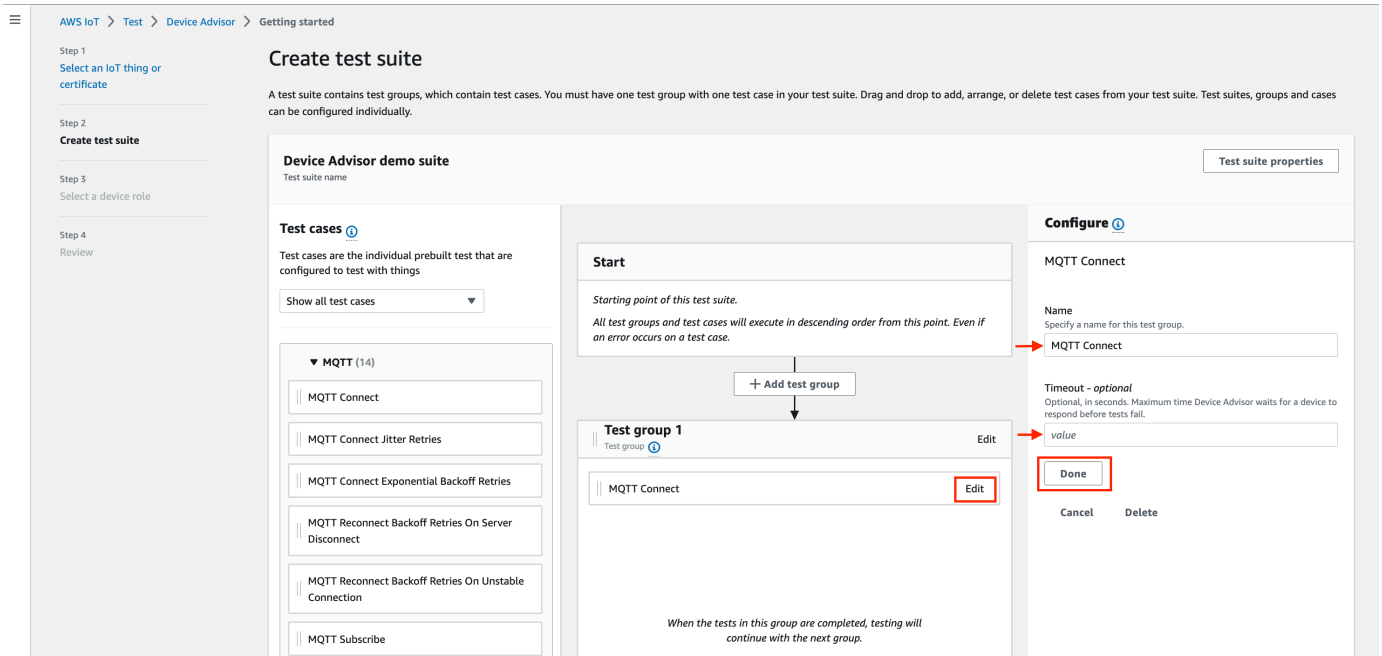
   Lorsque vous avez terminé, choisissez Done (Terminé) pour continuer.



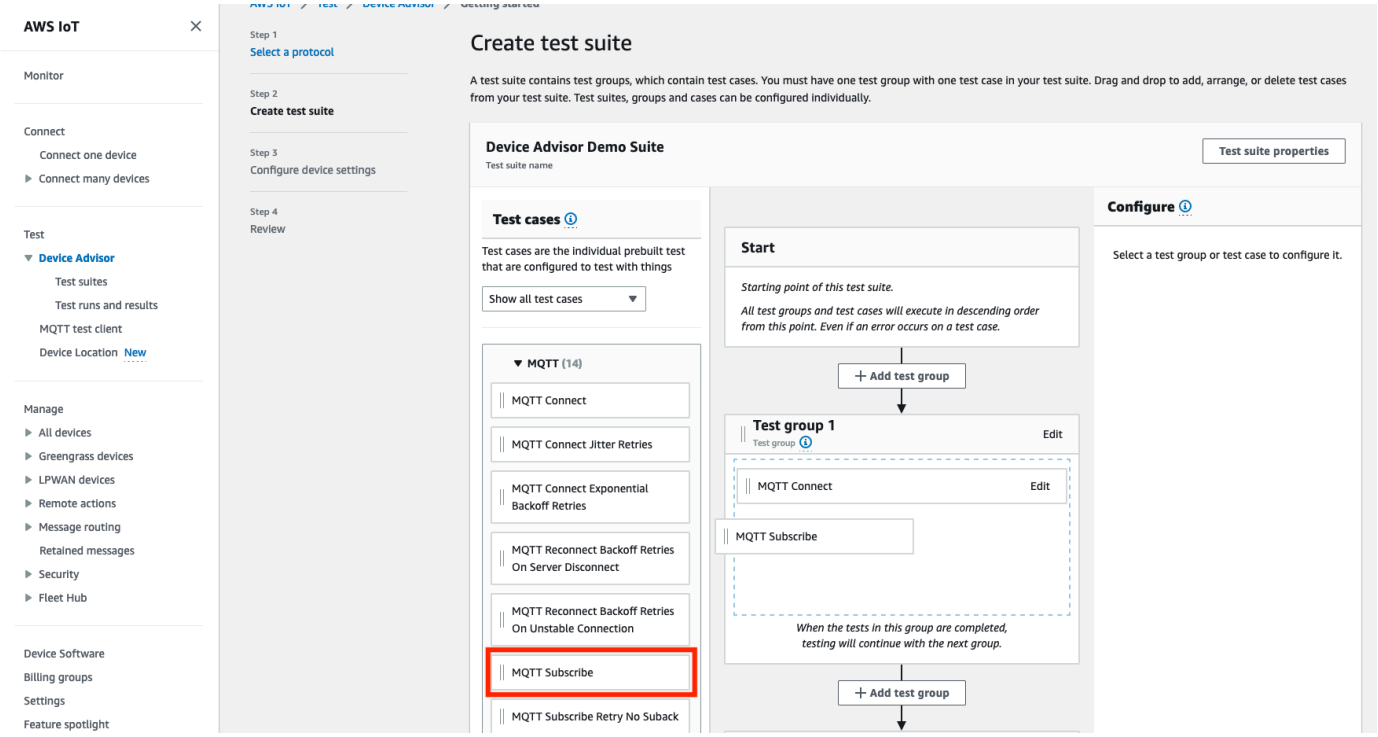
6. (Facultatif) Pour mettre à jour la configuration d'un scénario de test, choisissez le bouton Edit (Modifier) en regard du nom du scénario de test.

   - Name (Nom) : Entrez un nom personnalisé pour le groupe de suites de tests.
   - Délai (facultatif) : délai d'expiration (en secondes) pour le scénario de test sélectionné. Si vous ne spécifiez pas de valeur de délai d'attente, la valeur par défaut est utilisée.

   Lorsque vous avez terminé, choisissez Done (Terminé) pour continuer.

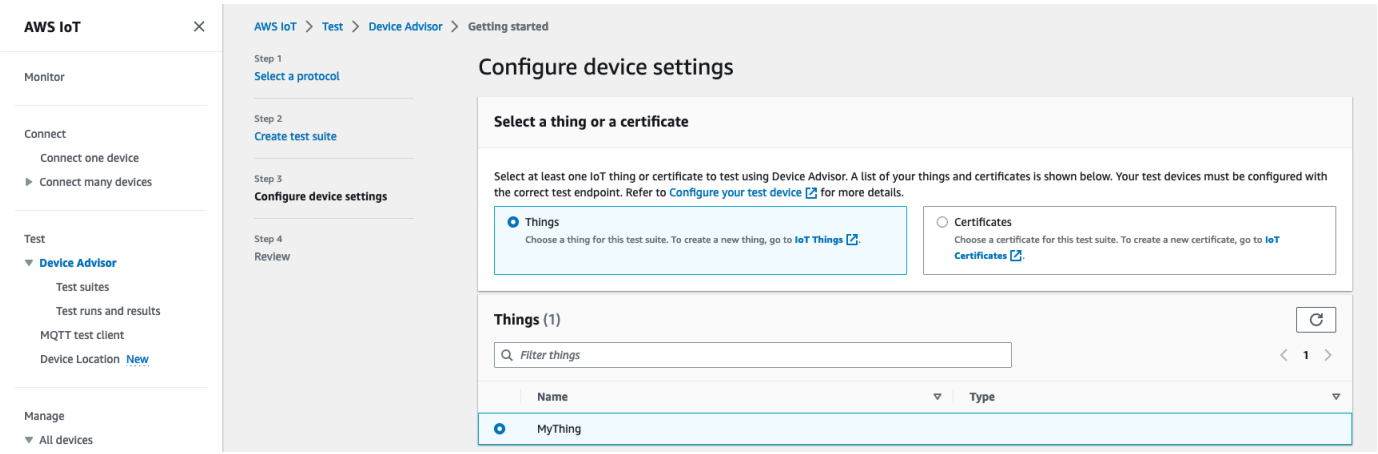


7. (Facultatif) Pour ajouter d'autres groupes de tests à la suite de tests, choisissez Ajouter un groupe de test, puis suivez les instructions de l'étape 5.
8. (Facultatif) Pour ajouter d'autres scénarios de test, faites glisser les cas de test de la section Cas de test vers l'un de vos groupes de tests.



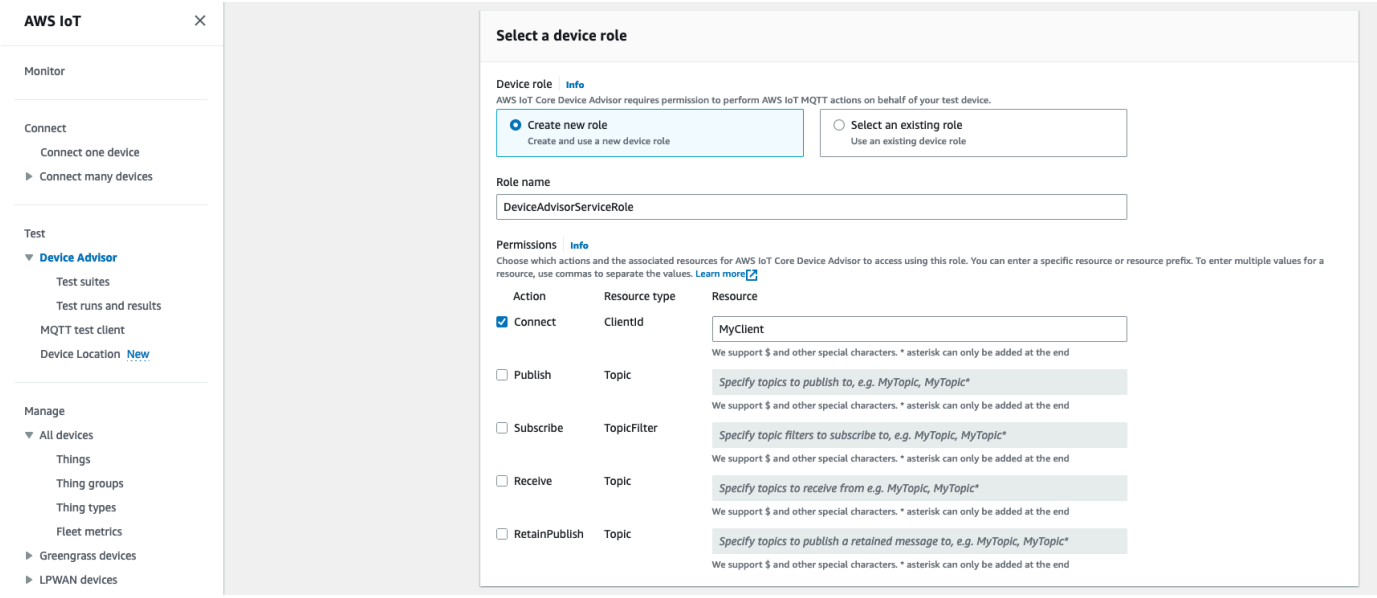
9. Vous pouvez modifier l'ordre de vos groupes de test et de vos scénarios de test. Pour apporter des modifications, faites glisser les scénarios de test répertoriés vers le haut ou vers le bas de la liste. Device Advisor exécute les tests dans l'ordre dans lequel vous les avez listés.

   Après avoir configuré votre suite de tests, choisissez Next.
10. À l'étape 3, sélectionnez un AWS IoT objet ou un certificat à tester à l'aide de Device Advisor. Si vous n'avez aucun élément ou certificat existant, consultez la section Configuration.

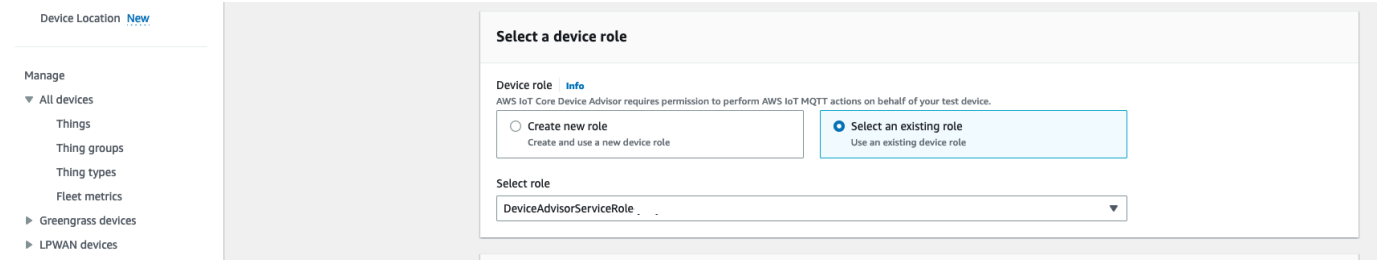


11. Vous pouvez configurer un rôle d'appareil que Device Advisor utilise pour effectuer des actions AWS IoT MQTT au nom de votre périphérique de test. Pour le cas de test MQTT Connect uniquement, l'action Connect est sélectionnée automatiquement. Cela est dû au fait que le rôle d'appareil nécessite cette autorisation pour exécuter la suite de tests. Pour les autres cas de test, les actions correspondantes sont sélectionnées.

    Indiquez les valeurs des ressources pour chacune des actions sélectionnées. Par exemple, pour l'action Connect, indiquez l'ID client que votre appareil utilise pour se connecter au point de terminaison Device Advisor. Vous pouvez fournir plusieurs valeurs séparées par des virgules et des valeurs de préfixe avec un caractère générique (*). Par exemple, pour autoriser la publication sur une rubrique commençant par `MyTopic`, entrez **`MyTopic*`** comme valeur de ressource.

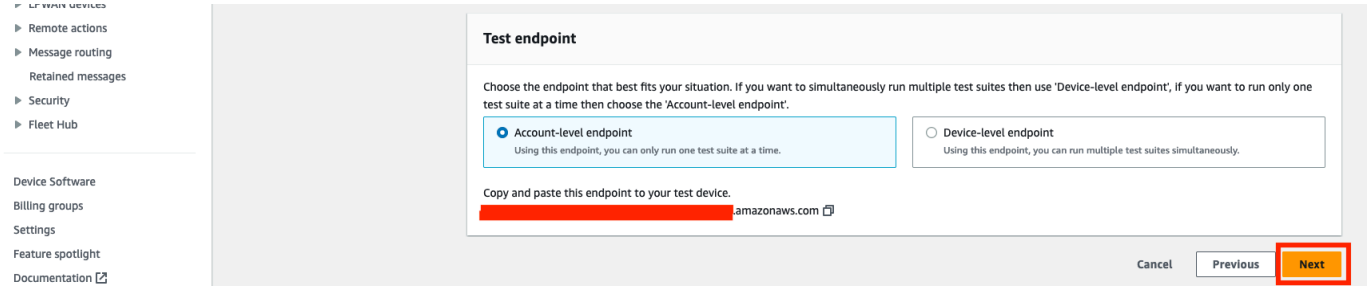


    Pour utiliser un rôle d'appareil créé précédemment dans Configuration, choisissez Sélectionner un rôle existant. Choisissez ensuite le rôle de votre appareil sous Sélectionner un rôle.



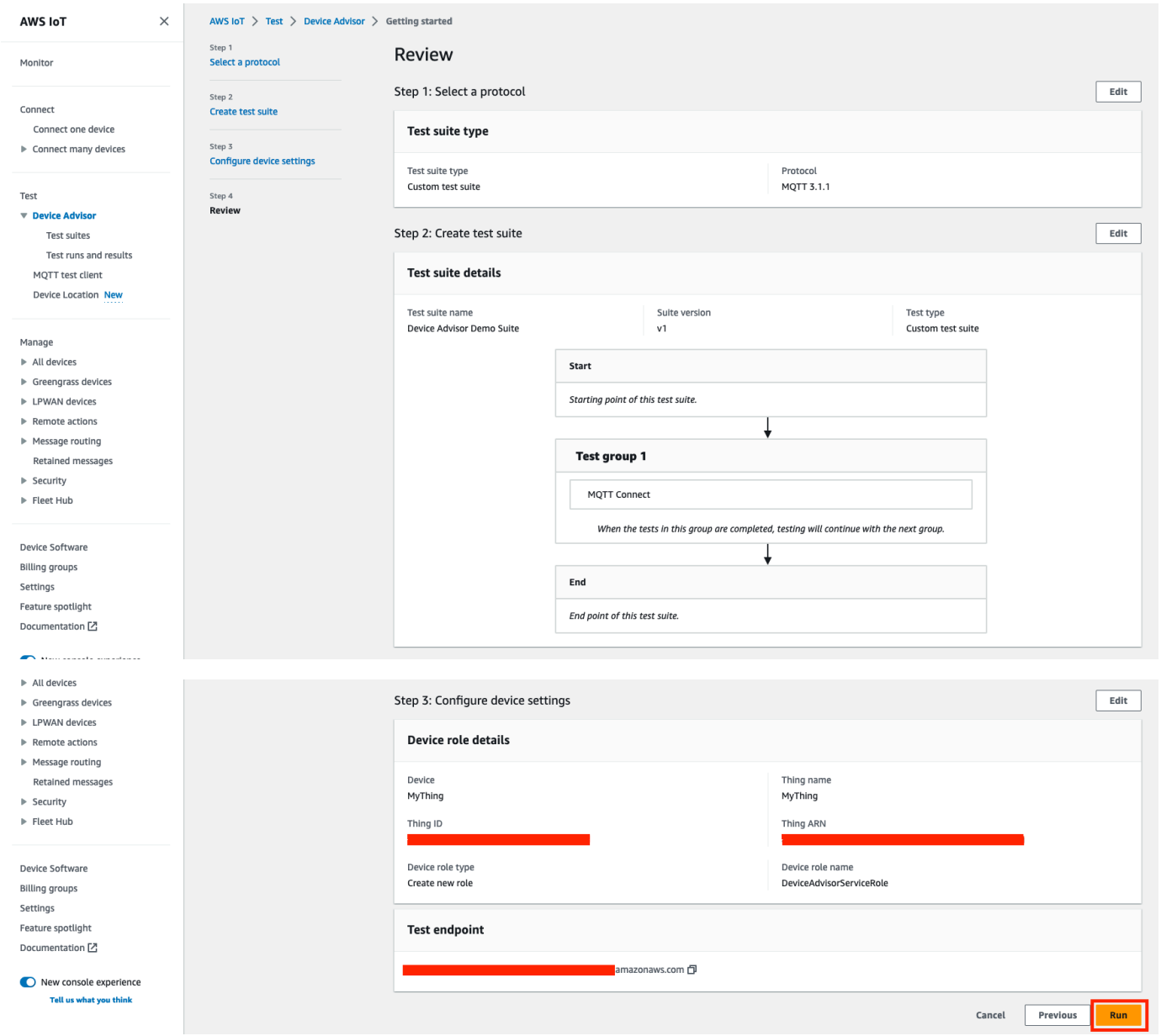
    Configurez le rôle de votre appareil à l'aide de l'une des deux options proposées, puis choisissez Next.

12. Dans la section Tester le point de terminaison, sélectionnez le point de terminaison qui convient le mieux à votre cas d'utilisation. Pour exécuter plusieurs suites de tests simultanément avec la même suite Compte AWS, sélectionnez le point de terminaison au niveau de l'appareil. Pour exécuter une suite de tests à la fois, sélectionnez Point de terminaison au niveau du compte.

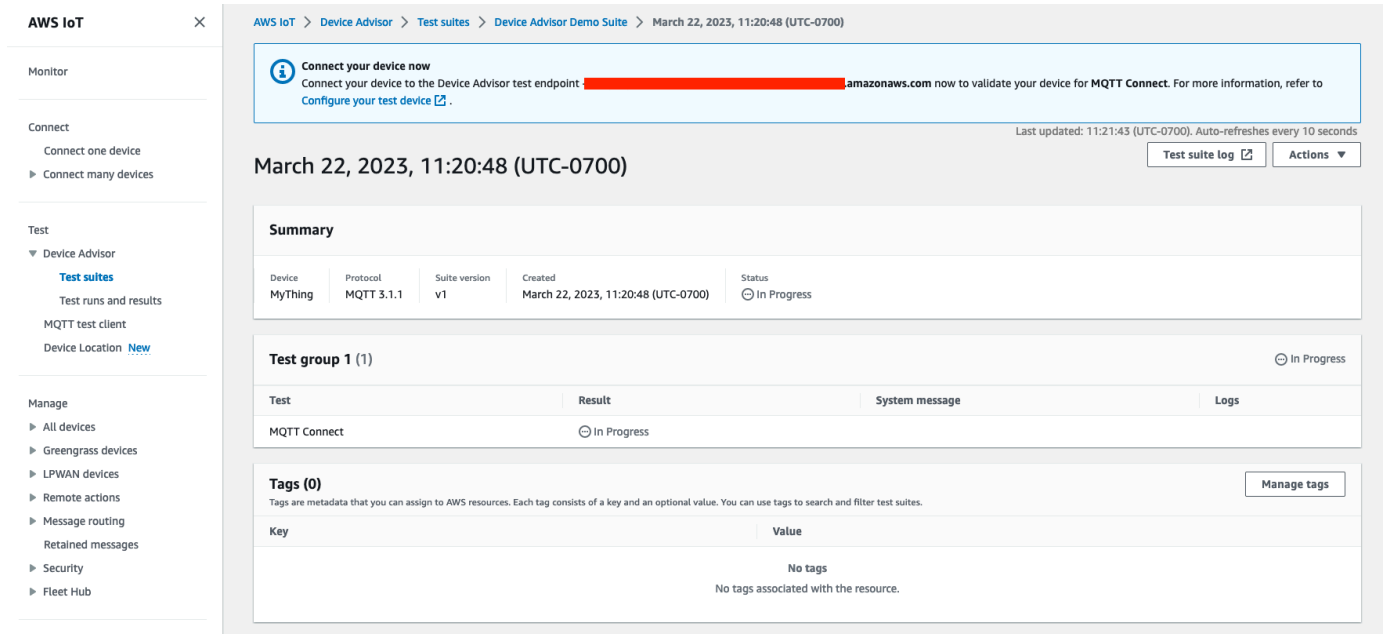


13. L'étape 4 présente une vue d'ensemble du périphérique de test sélectionné, du point de terminaison de test, de la suite de tests et du rôle d'appareil de test configurés. Pour apporter des modifications à une section, choisissez le bouton Edit (Modifier) correspondant à la section que vous souhaitez modifier. Une fois que vous avez confirmé votre configuration de test, choisissez Exécuter pour créer la suite de tests et exécuter vos tests.

> **Note**
>
> Pour de meilleurs résultats, vous pouvez connecter l'appareil de test sélectionné au point de terminaison de test Device Advisor avant de démarrer l'exécution de la suite de tests. Nous vous recommandons de créer un mécanisme permettant à votre appareil d'essayer de se connecter à notre point de terminaison de test toutes les cinq secondes pendant une à deux minutes au maximum.



14. Dans le volet de navigation, sous Test, choisissez Device Advisor, puis sélectionnez Test runs and results (Tests et résultats). Sélectionnez l'exécution d'une suite de tests pour afficher les détails de son exécution et ses journaux.



15. Pour accéder aux CloudWatch journaux Amazon de la suite, exécutez :

    - Choisissez Test suite log pour afficher les CloudWatch journaux relatifs à l'exécution de la suite de tests.
    - Choisissez Journal des cas de test pour n'importe quel cas de test afin d'afficher les CloudWatch journaux spécifiques au cas de test.

16. En fonction des résultats de vos tests, troubleshoot (dépannez) votre appareil jusqu'à ce que tous les tests réussissent.

Flux de travail Device Advisor

Ce didacticiel explique comment créer une suite de tests personnalisée et exécuter des tests sur le périphérique que vous souhaitez tester dans la console. Une fois les tests terminés, vous pouvez consulter les résultats et les journaux détaillés.

Prérequis

Avant de commencer ce didacticiel, suivez les étapes décrites dans Configuration.

Création d'une définition de suite de tests

Installez d'abord un AWS SDK.

Syntaxe de **`rootGroup`**

Un groupe racine est une chaîne JSON qui indique les cas de test à inclure dans votre suite de tests. Il spécifie également toutes les configurations nécessaires pour ces cas de test. Utilisez le groupe racine pour structurer et organiser votre suite de tests en fonction de vos besoins. La hiérarchie d'une suite de tests est la suivante :

```
test suite # test group(s) # test case(s)
```

Une suite de tests doit comporter au moins un groupe de test, et chaque groupe de test doit comporter au moins un cas de test. Device Advisor exécute les tests dans l'ordre dans lequel vous définissez les groupes de tests et les scénarios de test.

Chaque groupe racine suit cette structure de base :

```
{
    "configuration": {  // for all tests in the test suite
        "": ""
    }
    "tests": [{
        "name": ""
        "configuration": {  // for all sub-groups in this test group
            "": ""
        },
        "tests": [{
            "name": ""
            "configuration": {  // for all test cases in this test group
                "": ""
            },
            "test": {
                "id": ""
                "version": ""
            }
        }]
    }]
}
```

Dans le groupe racine, vous définissez la suite de tests avec un `nameconfiguration`, et le `tests` qui contient le groupe. Le groupe `tests` contient les définitions des tests individuels. Vous définissez chaque test avec un `nameconfiguration`, et un `test` bloc qui définit les cas de test pour ce test. Enfin, chaque cas de test est défini avec un `id` et `version`.

Pour plus d'informations sur l'utilisation des champs `"id"` et `"version"` pour chaque cas de test (`test` bloc), consultez Cas de test Device Advisor. Cette section contient également des informations sur les paramètres `configuration` disponibles.

Le bloc suivant est un exemple de configuration de groupe racine. Cette configuration spécifie les cas de test MQTT Connect Happy Case et MQTT Connect Exponential Backoff Retries, ainsi que les descriptions des champs de configuration.

```
{
    "configuration": {},  // Suite-level configuration
    "tests": [            // Group definitions should be provided here
      {
        "name": "My_MQTT_Connect_Group",  // Group definition name
        "configuration": {}               // Group definition-level configuration,
        "tests": [                        // Test case definitions should be provided
 here
        {
            "name": "My_MQTT_Connect_Happy_Case",  // Test case definition name
            "configuration": {
                "EXECUTION_TIMEOUT": 300        // Test case definition-level
 configuration, in seconds
            },
            "test": {
                "id": "MQTT_Connect",              // test case id
                "version": "0.0.0"                 // test case version
            }
        },
        {
            "name": "My_MQTT_Connect_Jitter_Backoff_Retries",  // Test case definition
 name
            "configuration": {
                "EXECUTION_TIMEOUT": 600                // Test case definition-level
 configuration,  in seconds
            },
            "test": {
                "id": "MQTT_Connect_Jitter_Backoff_Retries",  // test case id
                "version": "0.0.0"                                  // test case version
            }
        }]
    }]
}
```

Vous devez fournir la configuration du groupe racine lorsque vous créez la définition de votre suite de tests. Enregistrez le `suiteDefinitionId` renvoyé dans l'objet de réponse. Vous pouvez utiliser cet ID pour récupérer les informations de définition de votre suite de tests et exécuter votre suite de tests.

Voici un exemple de SDK Java :

```
response = iotDeviceAdvisorClient.createSuiteDefinition(
        CreateSuiteDefinitionRequest.builder()
            .suiteDefinitionConfiguration(SuiteDefinitionConfiguration.builder()
                .suiteDefinitionName("your-suite-definition-name")
                .devices(
                    DeviceUnderTest.builder()
                        .thingArn("your-test-device-thing-arn")
                        .certificateArn("your-test-device-certificate-arn")
                        .deviceRoleArn("your-device-role-arn") //if using SigV4 for
 MQTT over WebSocket
                        .build()
                )
                .rootGroup("your-root-group-configuration")
                .devicePermissionRoleArn("your-device-permission-role-arn")
                .protocol("MqttV3_1_1 || MqttV5 || MqttV3_1_1_OverWebSocket ||
 MqttV5_OverWebSocket")
                .build()
            )
            .build()
)
```

Obtenir une définition de suite de tests

Après avoir créé la définition de votre suite de tests, vous recevez `suiteDefinitionId` l'objet de réponse de l'opération d'API `CreateSuiteDefinition`.

Lorsque l'opération renvoie le `suiteDefinitionId`, vous pouvez voir de nouveaux champs `id` dans chaque groupe et une définition de cas de test dans le groupe racine. Vous pouvez les utiliser IDs pour exécuter un sous-ensemble de la définition de votre suite de tests.

Exemples de SDK Java

```
response = iotDeviceAdvisorClient.GetSuiteDefinition(
    GetSuiteDefinitionRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .build()
```

```
)
```

Obtenez un point de terminaison de test

Utilisez l'opération `GetEndpoint` API pour obtenir le point de terminaison de test utilisé par votre appareil. Sélectionnez le point de terminaison qui correspond le mieux à votre test. Pour exécuter simultanément plusieurs suites de tests, utilisez le point de terminaison au niveau de l'appareil en fournissant un `thing ARN`, `certificate ARN`, ou `device role ARN`. Pour exécuter une seule suite de tests, ne fournissez aucun argument à l' GetEndpoint opération permettant de choisir le point de terminaison au niveau du compte.

Exemples de kit SDK :

```
response = iotDeviceAdvisorClient.getEndpoint(GetEndpointRequest.builder()
.certificateArn("your-test-device-certificate-arn")
.thingArn("your-test-device-thing-arn")
.deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

.build())
```

Lancer l'exécution d'une suite de test

Après avoir créé une définition de suite de tests et configuré votre appareil de test pour qu'il se connecte à votre point de terminaison de test Device Advisor, exécutez votre suite de tests avec l'API `StartSuiteRun`.

Pour les clients MQTT, utilisez l'un ou l'autre `certificateArn` ou `thingArn` pour exécuter la suite de tests. Si les deux sont configurés, le certificat est utilisé s'il appartient à l'objet.

Pour MQTT over WebSocket customer, utilisez-le `deviceRoleArn` pour exécuter la suite de tests. Si le rôle spécifié est différent du rôle spécifié dans la définition de la suite de tests, le rôle spécifié remplace le rôle défini.

Pour `.parallelRun()`, utilisez `true` si vous utilisez un point de terminaison au niveau de l'appareil pour exécuter plusieurs suites de tests en parallèle en utilisant une seule Compte AWS.

Exemples de kit SDK :

```
response = iotDeviceAdvisorClient.startSuiteRun(StartSuiteRunRequest.builder()
.suiteDefinitionId("your-suite-definition-id")
```

```
.suiteRunConfiguration(SuiteRunConfiguration.builder()
    .primaryDevice(DeviceUnderTest.builder()
        .certificateArn("your-test-device-certificate-arn")
        .thingArn("your-test-device-thing-arn")
        .deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

        .build())
    .parallelRun(true | false)
    .build())
.build())
```

Enregistrez le `suiteRunId` de la réponse. Vous l'utiliserez pour récupérer les résultats de cette suite de tests exécutée.

Exécutez une suite de tests

Après avoir lancé l'exécution d'une suite de test, vous pouvez vérifier sa progression et ses résultats à l'aide de l'API `GetSuiteRun`.

Exemples de kit SDK :

```
// Using the SDK, call the GetSuiteRun API.

response = iotDeviceAdvisorClient.GetSuiteRun(
GetSuiteRunRequest.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Arrêter l'exécution d'une suite de tests

Pour arrêter l'exécution d'une suite de tests toujours en cours, vous pouvez appeler l'opération API `StopSuiteRun`. Une fois que vous avez appelé l'opération `StopSuiteRun`, le service lance le processus de nettoyage. Pendant que le service exécute le processus de nettoyage, la suite de test exécute des mises à jour de statut sur `Stopping`. Le processus de nettoyage peut prendre plusieurs minutes. Une fois le processus terminé, la suite de tests exécute des mises à jour de statut sur `Stopped`. Une fois qu'un cycle de test est complètement arrêté, vous pouvez lancer une autre suite de test. Vous pouvez vérifier régulièrement l'état d'exécution de la suite à l'aide de l'opération API `GetSuiteRun`, comme indiqué dans la section précédente.

Exemples de kit SDK :

```
// Using the SDK, call the StopSuiteRun API.

response = iotDeviceAdvisorClient.StopSuiteRun(
StopSuiteRun.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

Obtenez un rapport de qualification pour une exécution réussie de la suite de tests de qualification

Si vous exécutez une suite de tests de qualification qui se termine avec succès, vous pouvez récupérer un rapport de qualification avec l'opération d'API `GetSuiteRunReport`. Vous utilisez ce rapport de qualification pour qualifier votre appareil dans le cadre du programme AWS IoT Core de qualification. Pour déterminer si votre suite de tests est une suite de tests de qualification, vérifiez si le paramètre `intendedForQualification` est défini sur `true`. Après avoir appelé l'opération API `GetSuiteRunReport`, vous pouvez télécharger le rapport à partir de l'URL renvoyée pendant 90 secondes au maximum. Si plus de 90 secondes se sont écoulées depuis la dernière fois que vous avez appelé l'opération `GetSuiteRunReport`, appelez-la à nouveau pour récupérer une nouvelle URL valide.

Exemples de kit SDK :

```
// Using the SDK, call the getSuiteRunReport API.

response = iotDeviceAdvisorClient.getSuiteRunReport(
    GetSuiteRunReportRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .suiteRunId("your-suite-run-id")
        .build()
)
```

Flux de travail détaillé sur la console Device Advisor

Dans ce didacticiel, vous allez créer une suite de tests personnalisée et exécuter des tests sur l'appareil que vous souhaitez tester dans la console. Une fois les tests terminés, vous pouvez consulter les résultats et les journaux détaillés.

Didacticiels

- Prérequis
- Création d'une définition de suite de tests
- Lancer l'exécution d'une suite de test
- Arrêter l'exécution d'une suite de tests (facultatif)
- Afficher les détails et les journaux d'exécution de la suite de tests
- Téléchargez un rapport AWS IoT de qualification

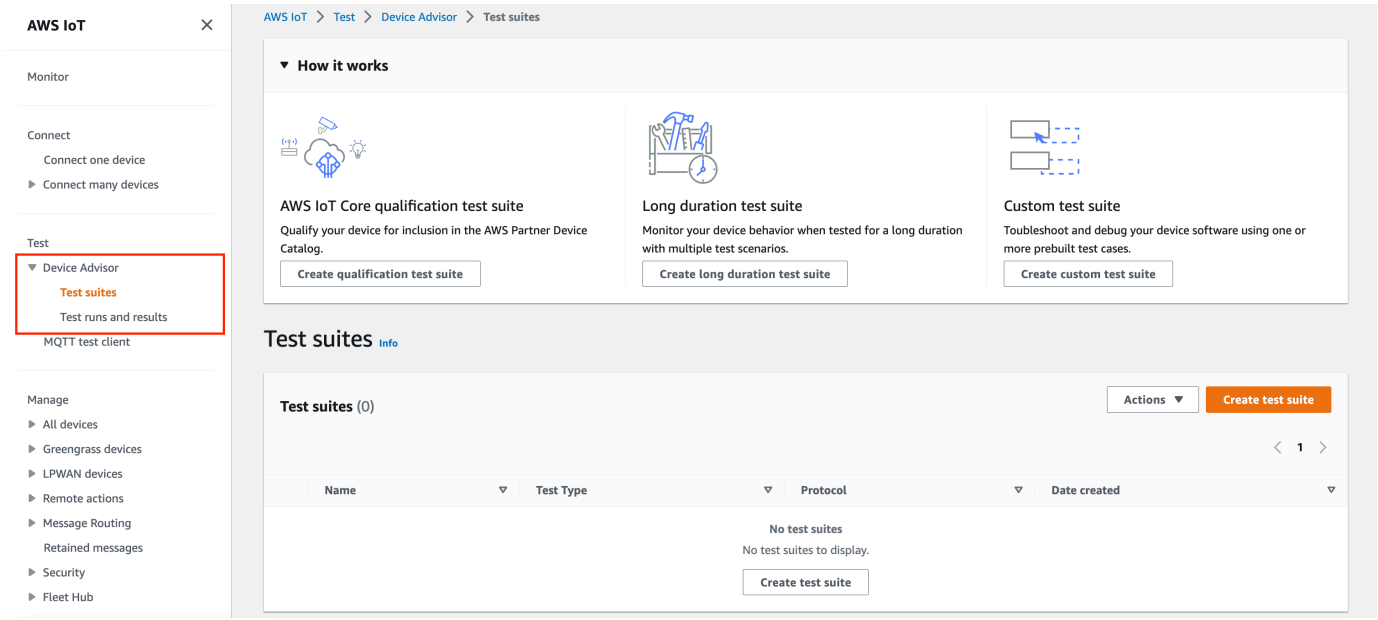
Prérequis

Pour effectuer ce didacticiel, vous devez créer un objet et un certificat.

Création d'une définition de suite de tests

Créez une suite de tests afin de pouvoir l'exécuter sur vos appareils et effectuer la vérification.

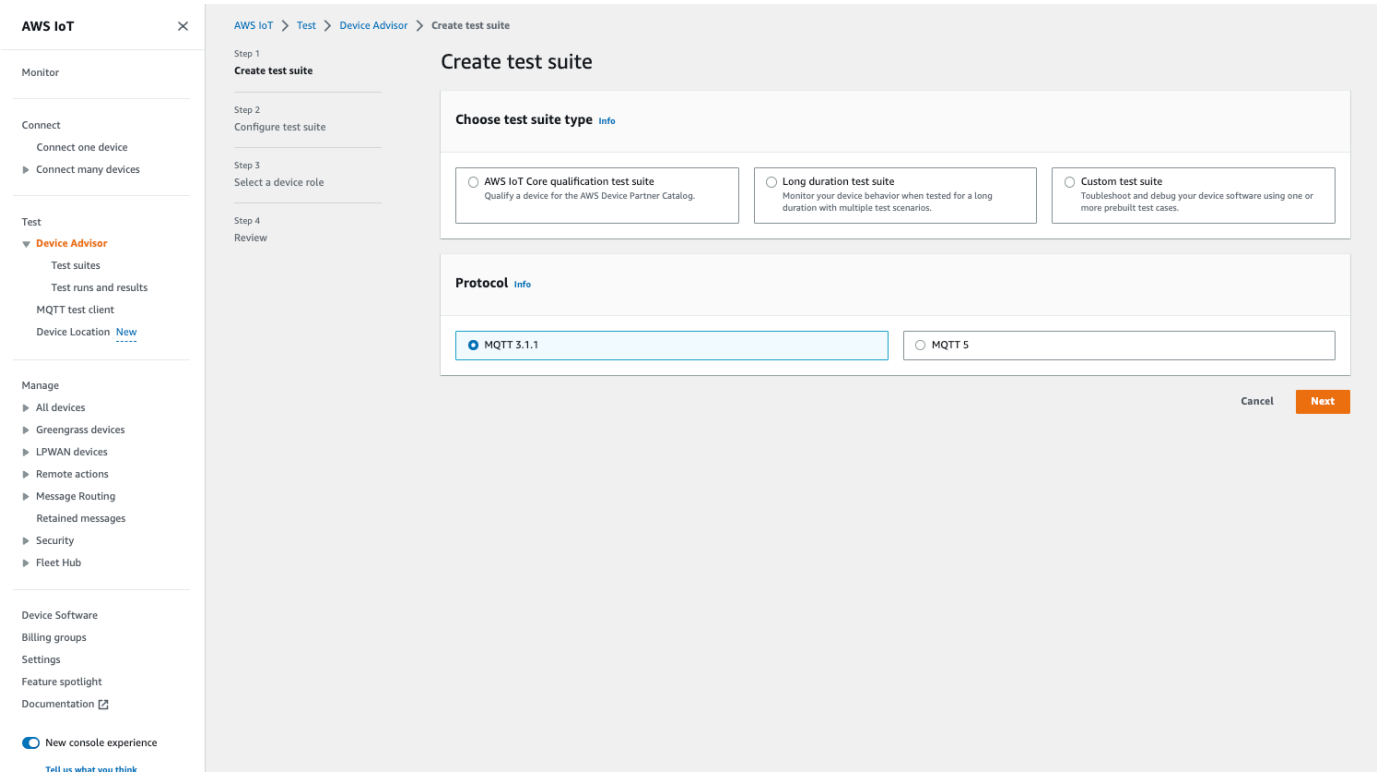
1. Dans la AWS IoT console, dans le volet de navigation, développez Test, Device Advisor, puis choisissez Test suites.



   Choisissez Create Test Suite.

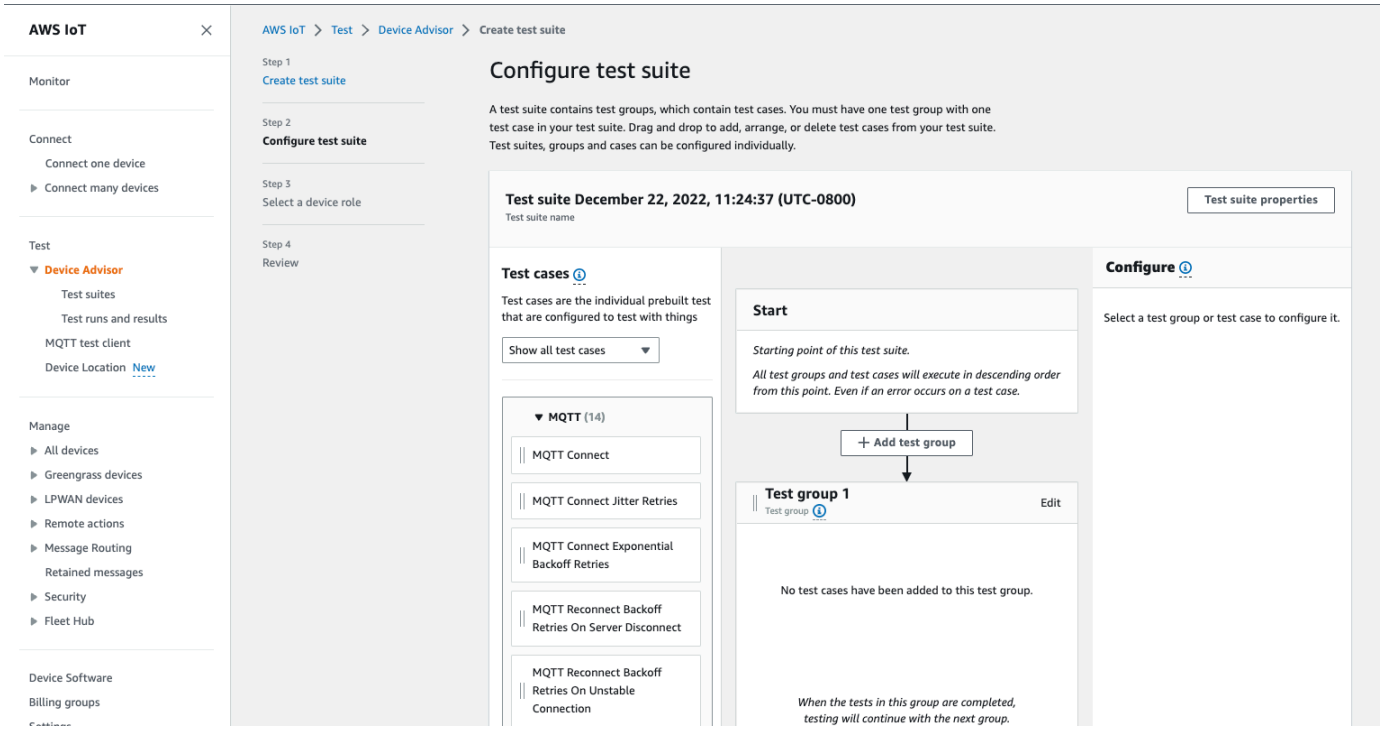
2. Sélectionnez `Use the AWS Qualification test suite` ou `Create a new test suite`.

   Pour le protocole, choisissez MQTT 3.1.1 ou MQTT 5.

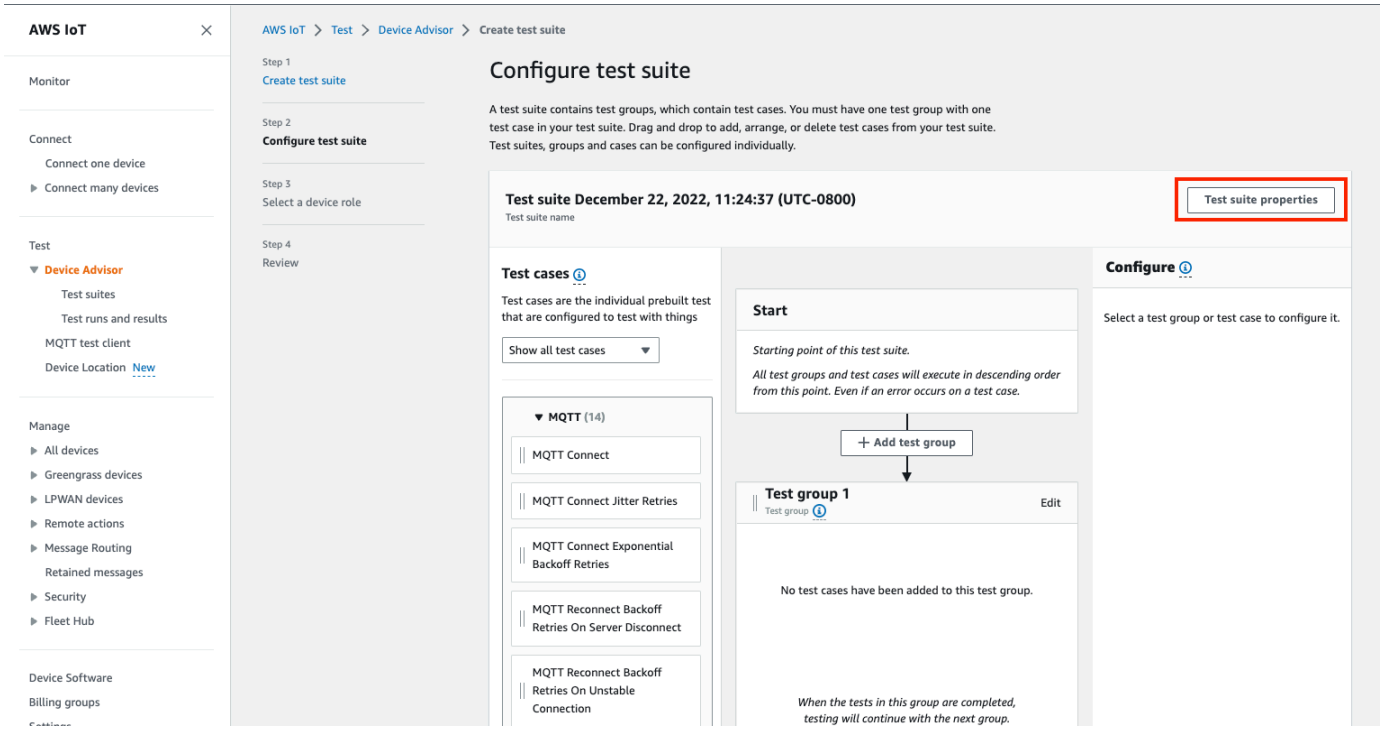


Sélectionnez `Use the AWS Qualification test suite` cette option pour qualifier et inscrire votre appareil dans le catalogue des appareils AWS partenaires. En choisissant cette option, les scénarios de test requis pour la qualification de votre appareil dans le cadre du programme de qualification AWS IoT Core sont présélectionnés. Les groupes de test et les scénarios de test ne peuvent pas être ajoutés ou supprimés. Vous devrez tout de même configurer les propriétés de la suite de tests.

Sélectionnez `Create a new test suite` pour créer et configurer une suite de test personnalisée. Nous vous recommandons de commencer par cette option pour les tests initiaux et le dépannage. Une suite de tests personnalisée doit comporter au moins un groupe de test, et chaque groupe de test doit comporter au moins un cas de test. Dans le cadre de ce didacticiel, nous allons sélectionner cette option et choisir Next.



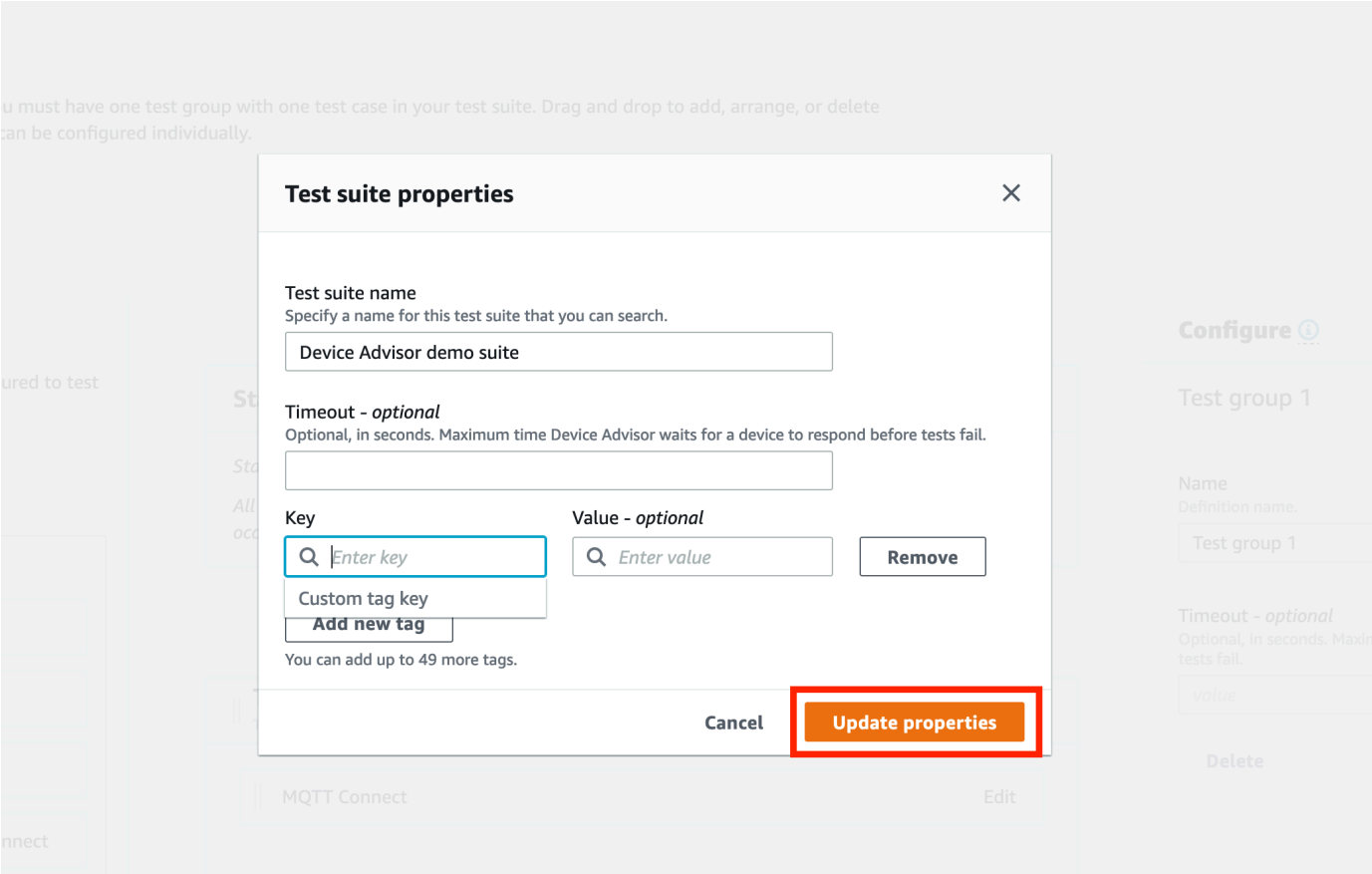
3. Choisissez Propriétés de la suite de tests. Vous devez créer les propriétés de la suite de tests lorsque vous créez votre suite de tests.



Sous Propriétés de la suite de tests, renseignez les champs suivants :

- Nom de la suite de test : vous pouvez créer la suite avec un nom personnalisé.

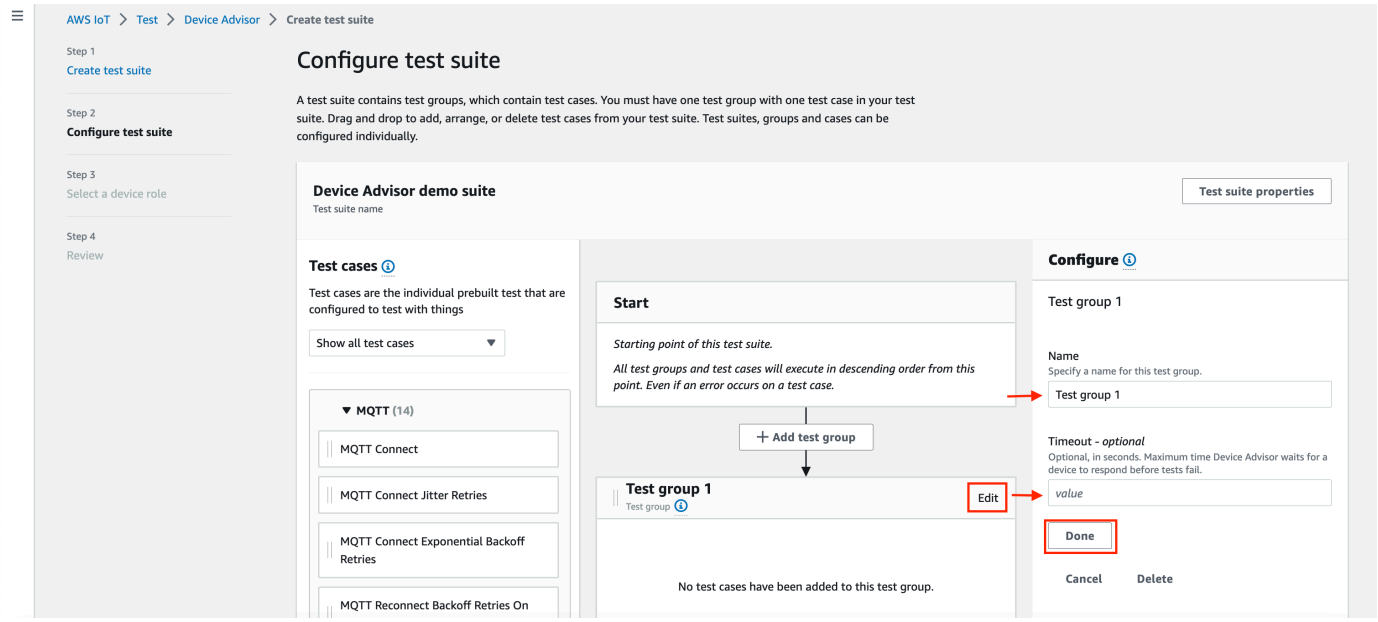
- Timeout (optional) Délai(facultatif) délai d'expiration (en secondes) pour chaque scénario de test de la suite de tests actuelle. Si vous ne spécifiez pas de valeur de délai d'attente, la valeur par défaut est utilisée.
- Balises (facultatif) : ajoutez des balises à la suite de tests.



Lorsque vous avez terminé, choisissez Mettre à jour les propriétés.

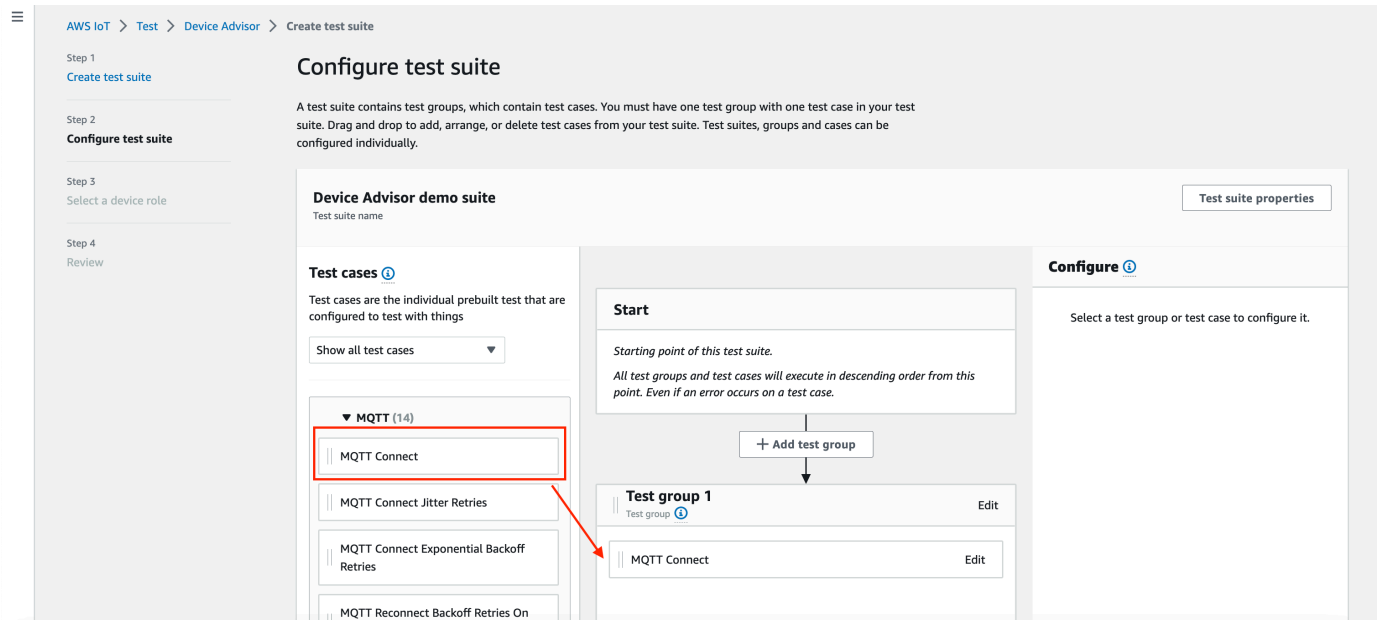
4. Pour modifier la configuration au niveau du groupe, sous `Test group 1`, choisissez Modifier. Entrez ensuite un nom pour donner un nom personnalisé au groupe.

   Facultativement, vous pouvez également saisir une valeur de délai d'expiration en secondes dans le groupe de test sélectionné. Si vous ne spécifiez pas de valeur de délai d'attente, la valeur par défaut est utilisée.



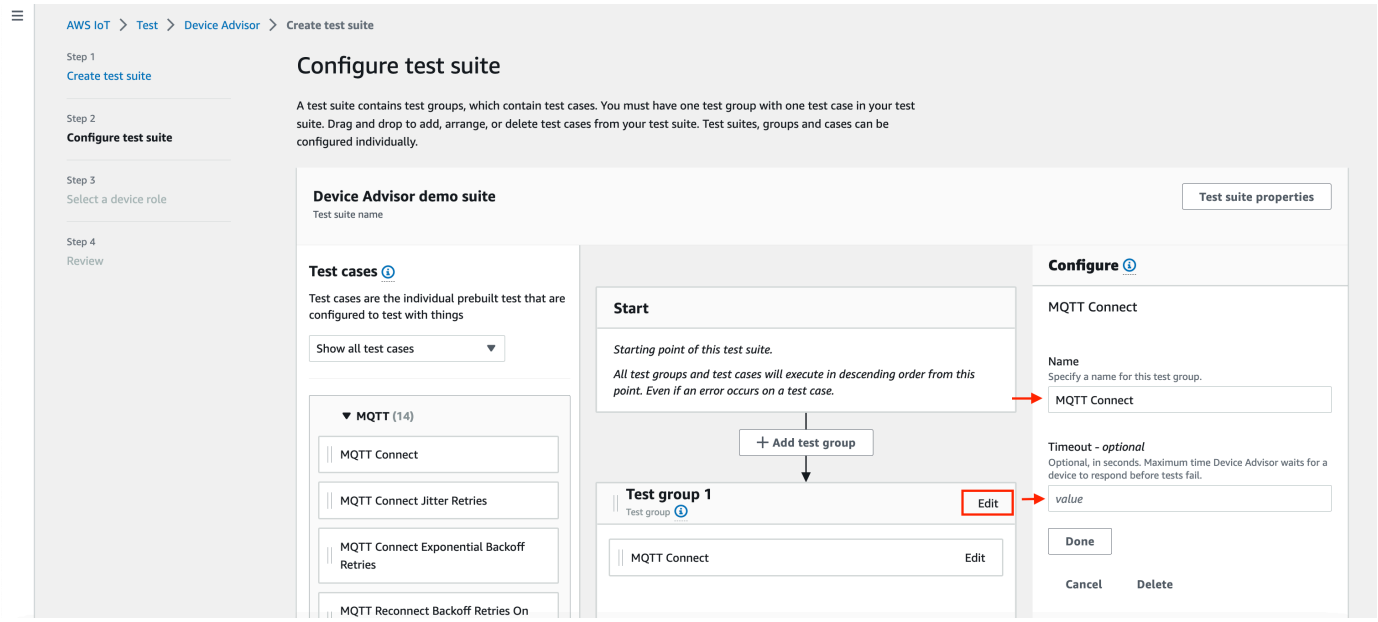
Sélectionnez Exécuté.

5. Faites glisser l'un des scénarios de test disponibles depuis les scénarios de test vers le groupe de test.



6. Pour modifier la configuration au niveau du scénario de test que vous avez ajouté à votre groupe de test, choisissez Modifier. Entrez ensuite un nom pour donner un nom personnalisé au groupe.

   Facultativement, vous pouvez également saisir une valeur de délai d'expiration en secondes dans le groupe de test sélectionné. Si vous ne spécifiez pas de valeur de délai d'attente, la valeur par défaut est utilisée.



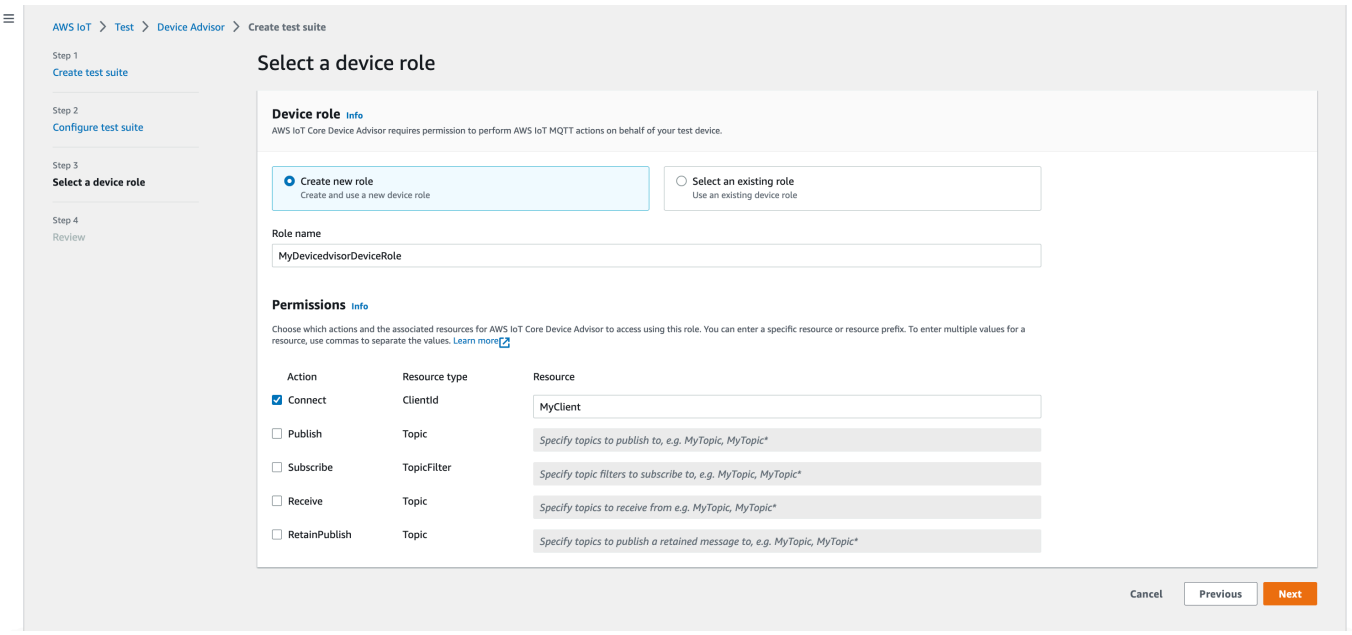
Sélectionnez Exécuté.

> **Note**
>
> Pour ajouter d'autres groupes de tests à la suite de tests, choisissez Ajouter un groupe de test. Suivez les étapes précédentes pour créer et configurer d'autres groupes de test, ou pour ajouter d'autres cas de test à un ou plusieurs groupes de test. Les groupes de tests et les scénarios de test peuvent être réorganisés en choisissant un scénario de test et en le faisant glisser vers la position souhaitée. Device Advisor exécute les tests dans l'ordre dans lequel vous définissez les groupes de tests et les scénarios de test.

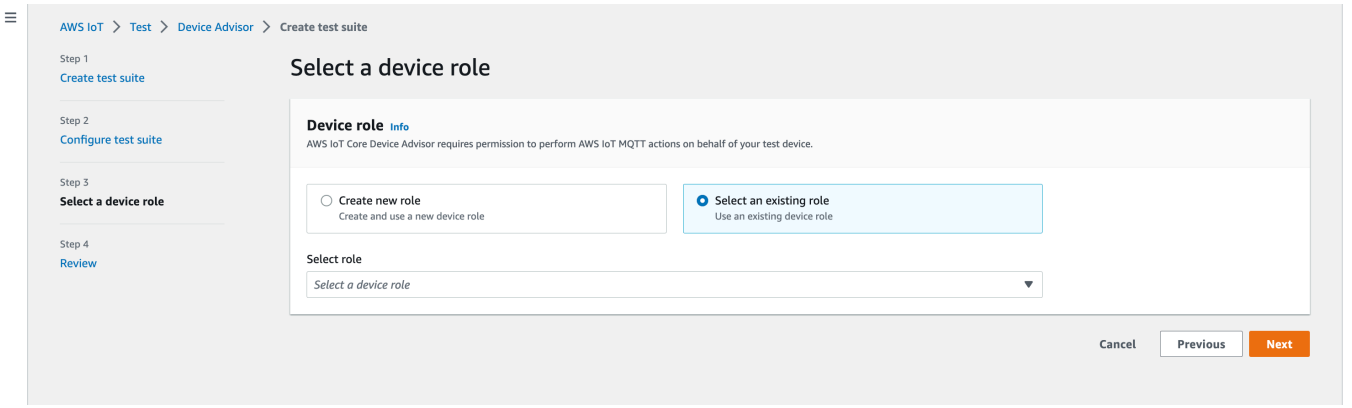
7. Choisissez Suivant.
8. À l'étape 3, configurez un rôle de périphérique que Device Advisor utilisera pour effectuer des actions AWS IoT MQTT au nom de votre périphérique de test.

   Si vous avez sélectionné le scénario de test MQTT Connect uniquement à l'étape 2, l'action Connect sera sera vérifiée automatiquement puisque cette autorisation est requise sur le rôle de périphérique pour exécuter cette suite de tests. Si vous avez sélectionné d'autres scénarios de test, les actions requises correspondantes seront vérifiées. Assurez-vous que les valeurs des ressources sont fournies pour chacune des actions. Par exemple, pour l'action Connect, indiquez l'ID client que votre appareil utilise pour se connecter au point de terminaison Device Advisor. Vous pouvez fournir plusieurs valeurs en utilisant des virgules pour séparer les valeurs, et vous pouvez également fournir des valeurs de préfixe en utilisant un caractère générique (*).

Par exemple, pour accorder l'autorisation de publier sur n'importe quel sujet commençant par `MyTopic`, vous pouvez fournir « `MyTopic*` » comme valeur de ressource.



Si vous avez déjà créé un rôle sur l'appareil et que vous souhaitez utiliser ce rôle, sélectionnez Sélectionner un rôle existant et choisissez votre rôle sur l'appareil sous Sélectionner un rôle.



Configurez le rôle de votre appareil à l'aide de l'une des deux options proposées et choisissez Next (Suivant).

9. À l'étape 4, assurez-vous que la configuration fournie à chacune des étapes est précise. Pour modifier la configuration fournie pour une étape particulière, choisissez Modifier pour l'étape correspondante.

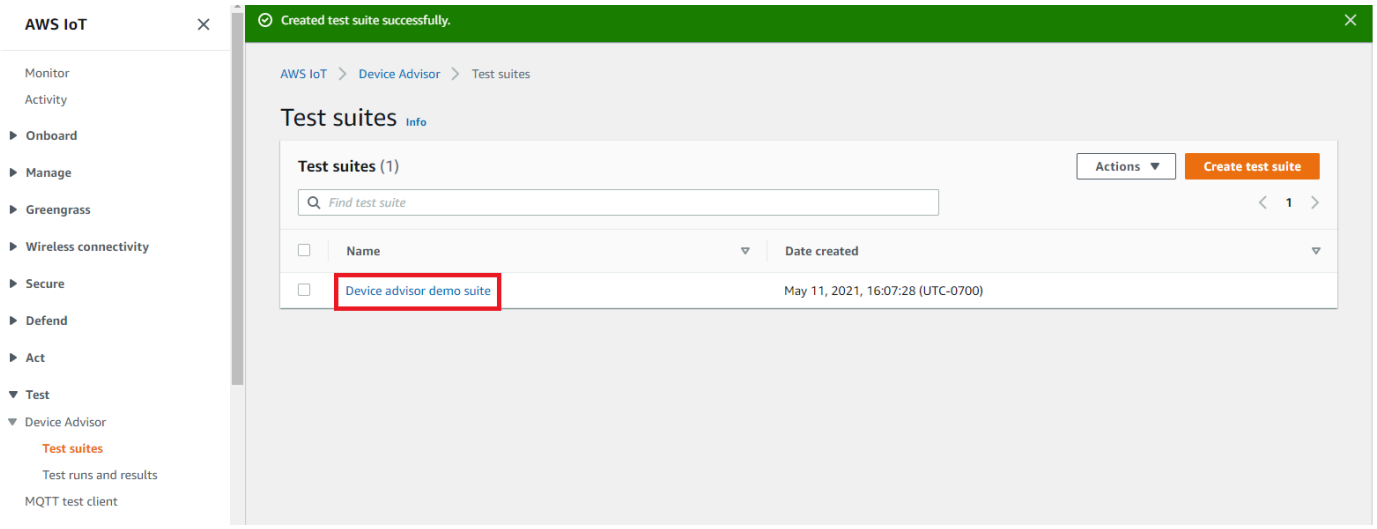
   Après avoir vérifié la configuration, choisissez Créer une suite de tests.

La suite de tests devrait être créée avec succès et vous serez redirigé vers la page des suites de tests où vous pourrez voir toutes les suites de tests créées.

Si la création de la suite de tests a échoué, assurez-vous que la suite de tests, les groupes de tests, les scénarios de test et le rôle de l'appareil ont été configurés conformément aux instructions précédentes.

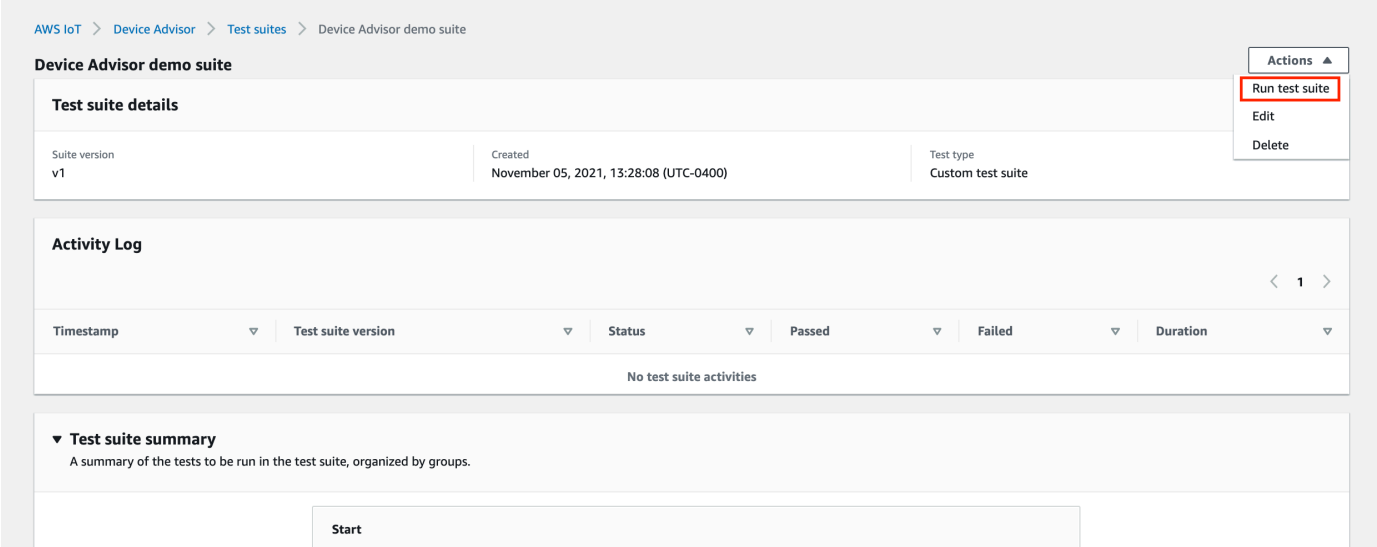
Lancer l'exécution d'une suite de test

1. Dans la AWS IoT console, dans le volet de navigation, développez Test, Device Advisor, puis choisissez Test suites.
2. Choisissez la suite de tests dont vous souhaitez consulter les détails.



La page détaillée de la suite de tests affiche toutes les informations relatives à la suite de tests.

3. Choisissez Actions, puis Exécuter la suite de tests.

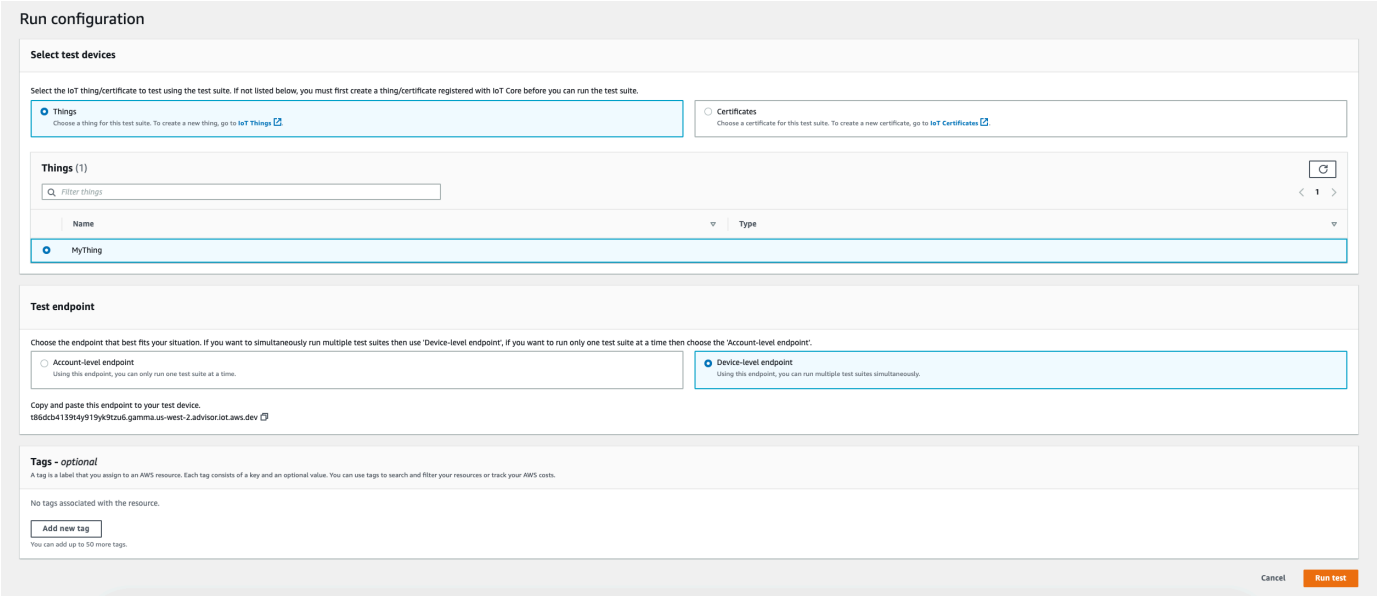


4. Sous Exécuter la configuration, vous devez sélectionner un AWS IoT objet ou un certificat à tester à l'aide de Device Advisor. Si vous n'avez aucun élément ou certificat existant, créez d'abord des AWS IoT Core ressources.

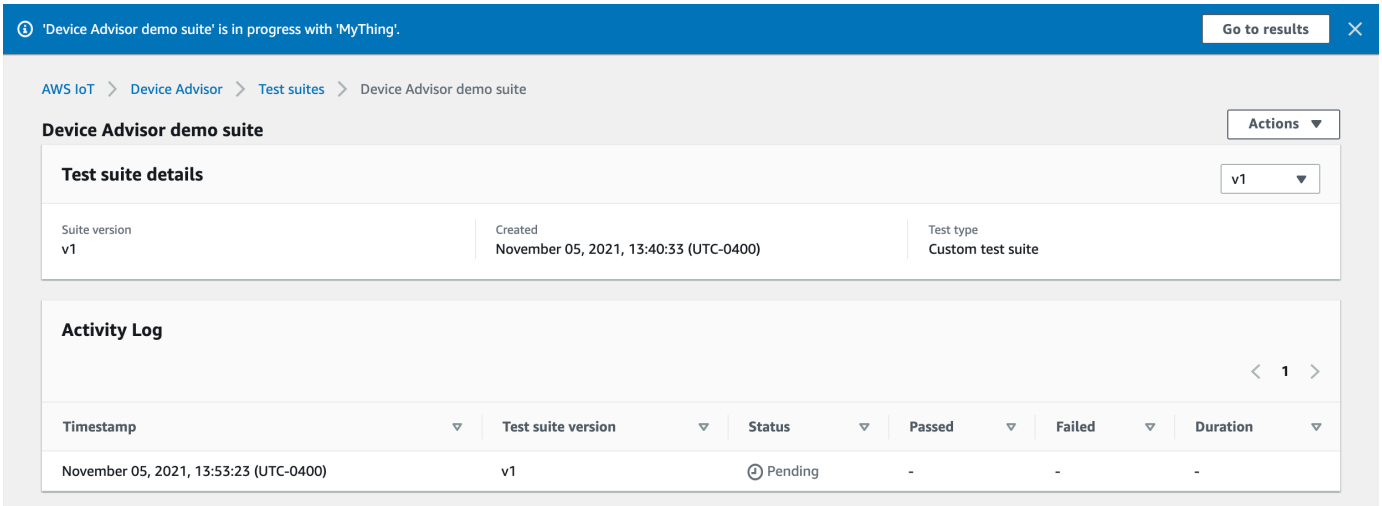
   Dans la section Tester le point de terminaison, sélectionnez le point de terminaison qui convient le mieux à votre cas d'utilisation. Si vous prévoyez d'exécuter plusieurs suites de tests simultanément en utilisant le même AWS compte à l'avenir, sélectionnez Endpoint au niveau de l'appareil. Sinon, si vous prévoyez d'exécuter une seule suite de tests à la fois, sélectionnez Point de terminaison au niveau du compte.

   Configurez votre appareil de test avec le point de terminaison de test du Device Advisor sélectionné.

   Après avoir sélectionné un objet ou un certificat et choisi un point de terminaison Device Advisor, choisissez Exécuter le test.

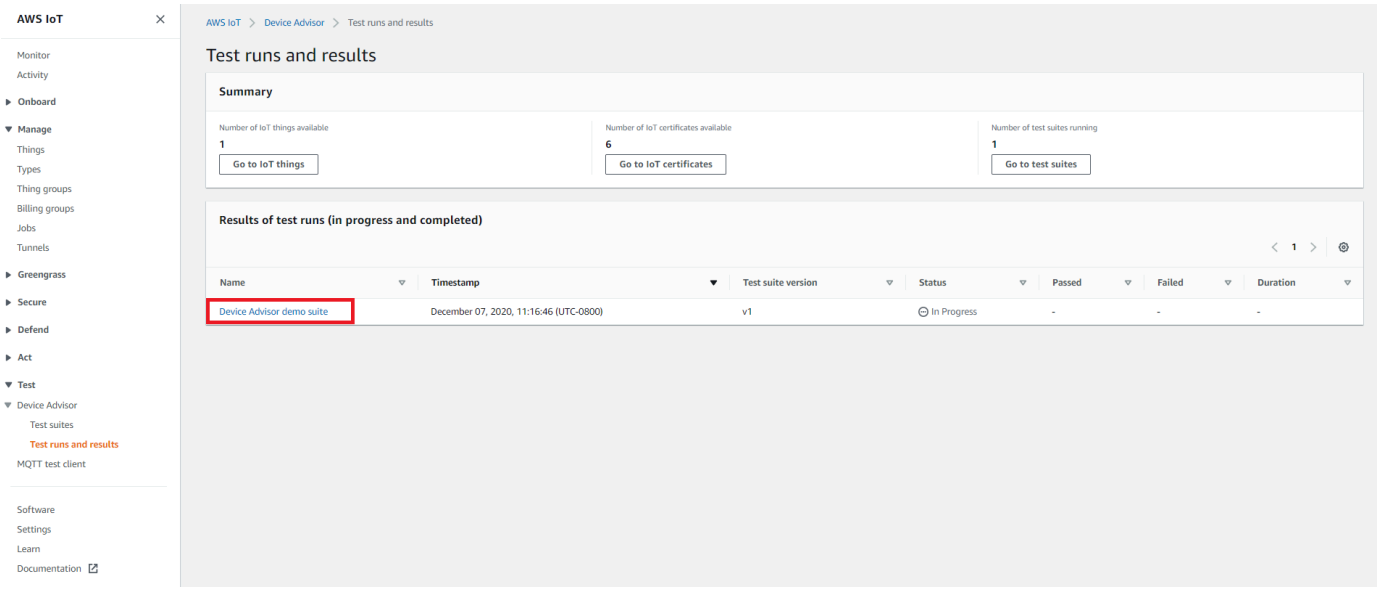


5. Choisissez Accéder aux résultats sur le bandeau supérieur pour afficher les détails du test.

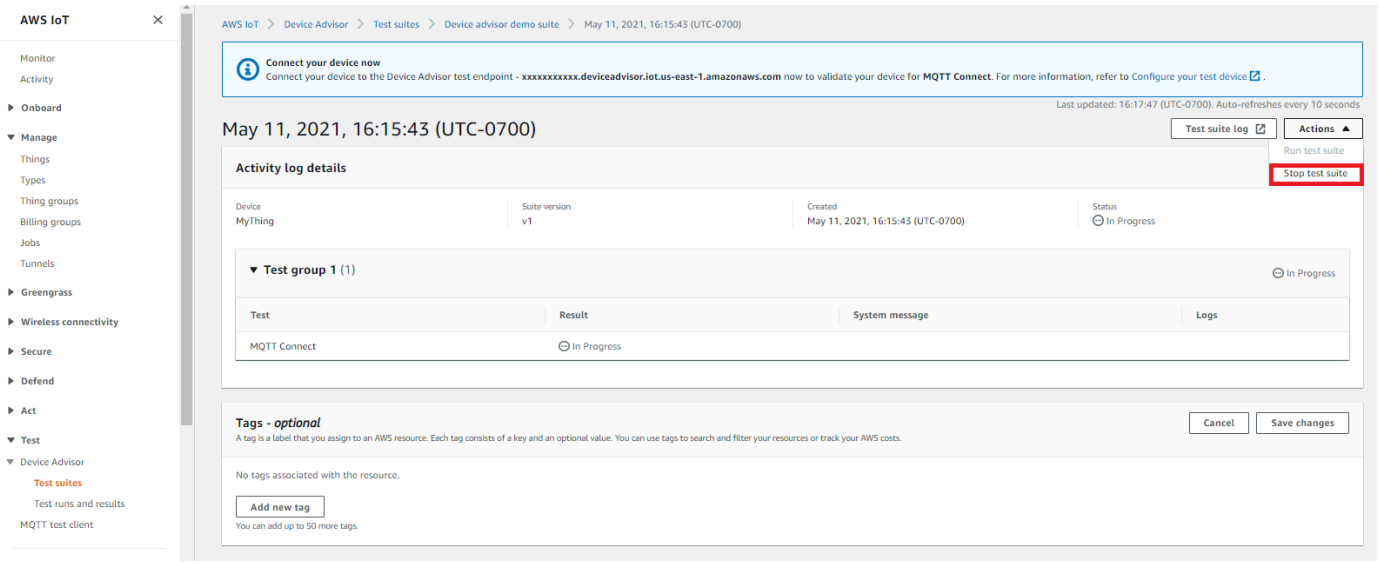


Arrêter l'exécution d'une suite de tests (facultatif)

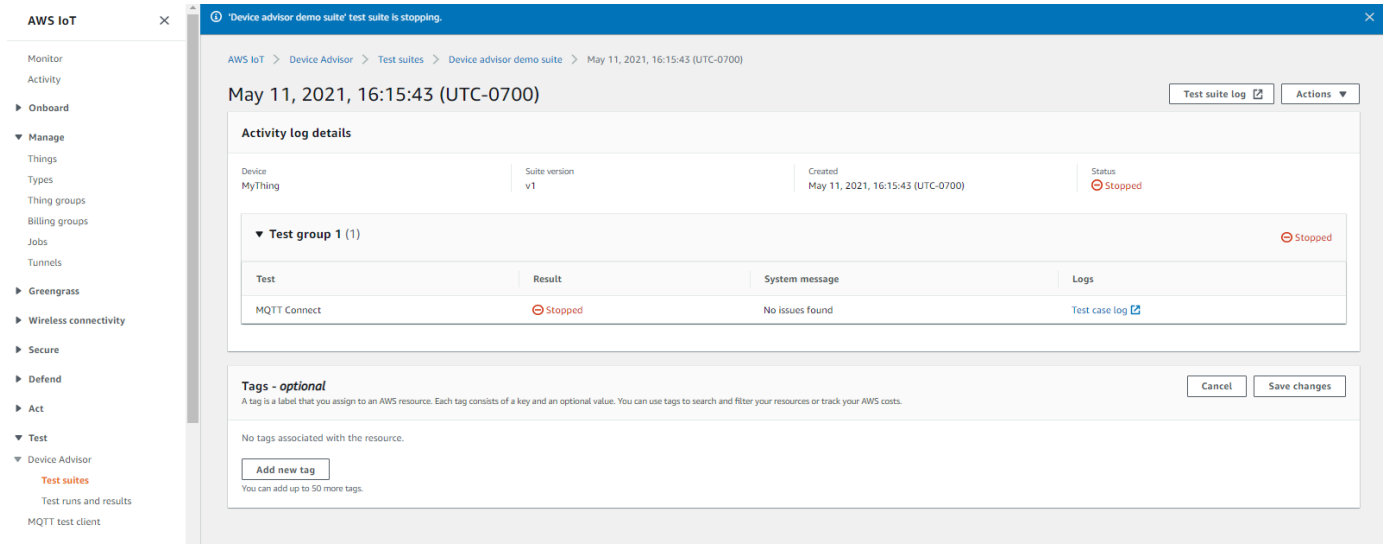
1. Dans la AWS IoT console, dans le volet de navigation, développez Test, Device Advisor, puis choisissez Tests et résultats.
2. Choisissez la suite de tests en cours que vous souhaitez arrêter.



3. Choisissez Actions, puis Arrêter la suite de tests.



4. Le processus de nettoyage prendra plusieurs minutes. Pendant l'exécution du processus de nettoyage, l'état du test sera `STOPPING`. Attendez que le processus de nettoyage soit terminé et que l'état de la suite de tests passe au `STOPPED` statut actuel avant de démarrer une nouvelle exécution de suite.

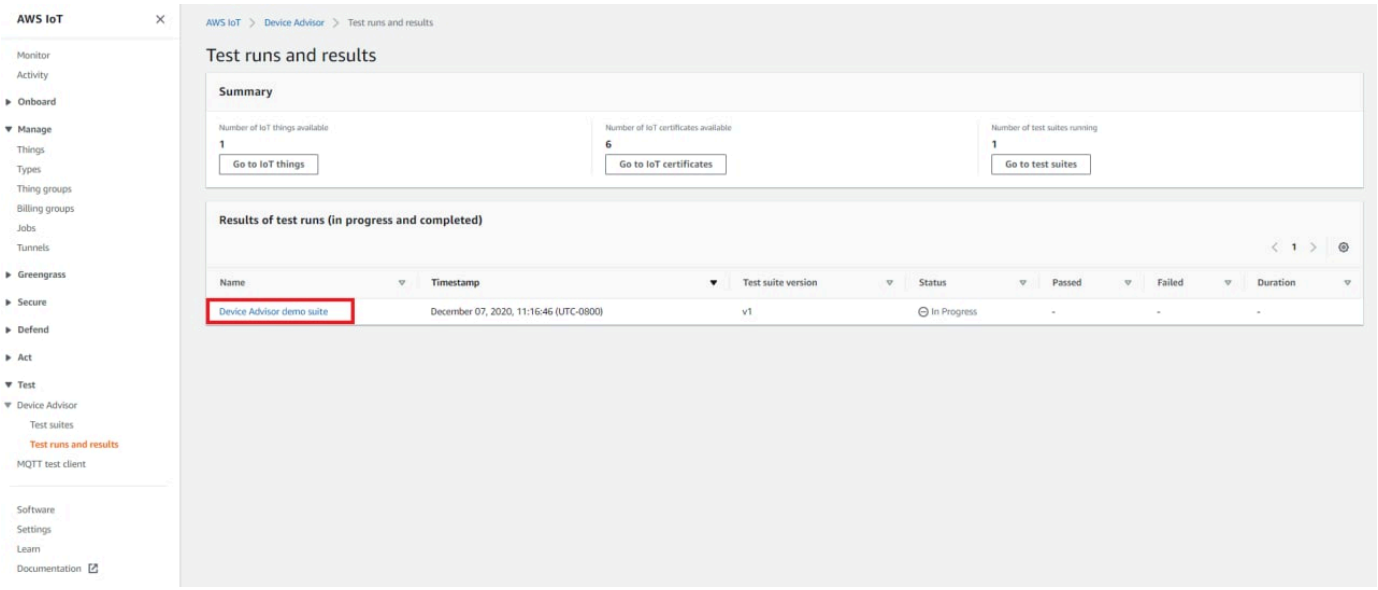


Afficher les détails et les journaux d'exécution de la suite de tests

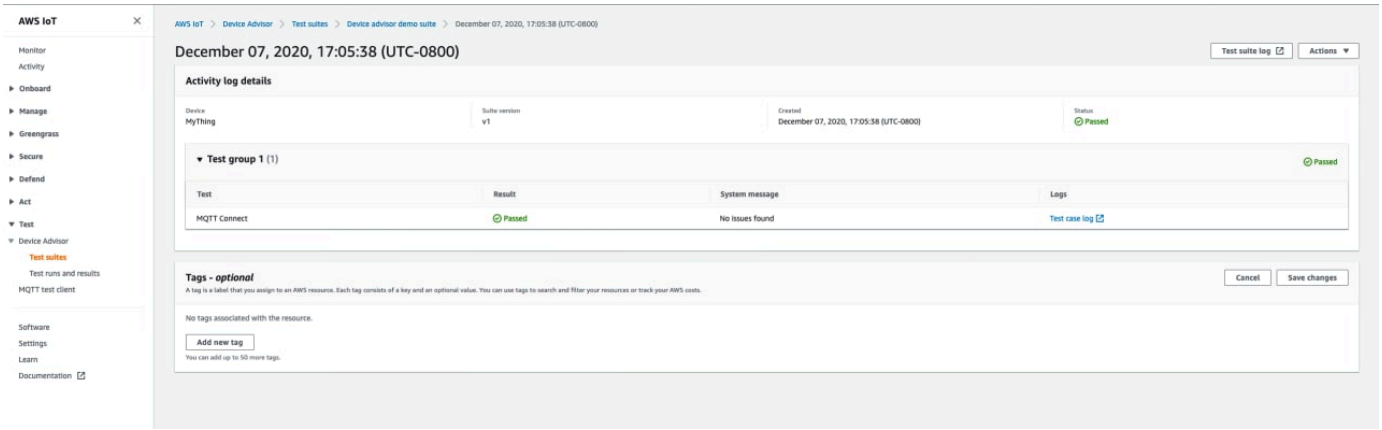
1. Dans la AWS IoT console, dans le volet de navigation, développez Test, Device Advisor, puis choisissez Tests et résultats.

   Cette page affiche :

   - Nombre d'objets IoT
   - Nombre de certificats IoT
   - Nombre de suites de tests en cours d'exécution
   - Toutes les suites de tests exécutées qui ont été créées
2. Choisissez la suite de tests pour laquelle vous souhaitez afficher les détails et les journaux d'exécution.



La page de résumé de l'exécution affiche l'état de l'exécution de la suite de tests en cours. Cette page est actualisée automatiquement toutes les 10 secondes. Nous vous recommandons de créer un mécanisme permettant à votre appareil d'essayer de se connecter à notre point de terminaison de test toutes les cinq secondes pendant une à deux minutes au maximum. Vous pouvez ensuite exécuter plusieurs scénarios de test en séquence de manière automatisée.



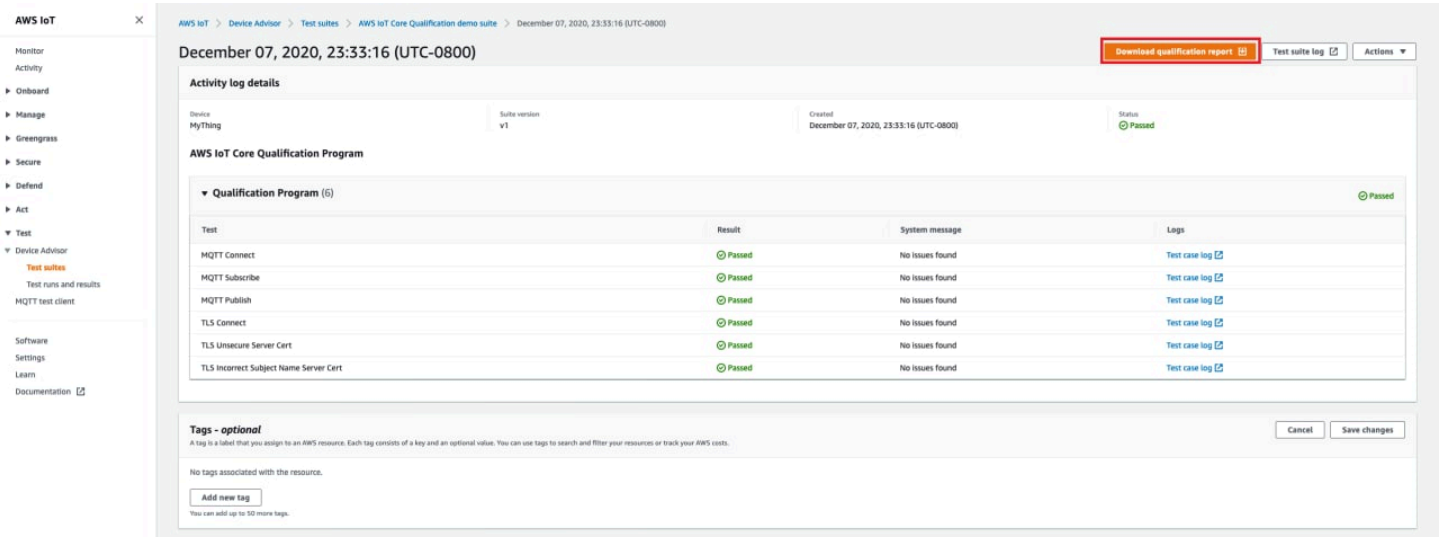
3. Pour accéder aux CloudWatch journaux relatifs à l'exécution de la suite de tests, choisissez Test suite log.

   Pour accéder aux CloudWatch journaux de n'importe quel scénario de test, choisissez Test case log.

4. En fonction des résultats de vos tests, troubleshoot (dépannez) votre appareil jusqu'à ce que tous les tests réussissent.

Téléchargez un rapport AWS IoT de qualification

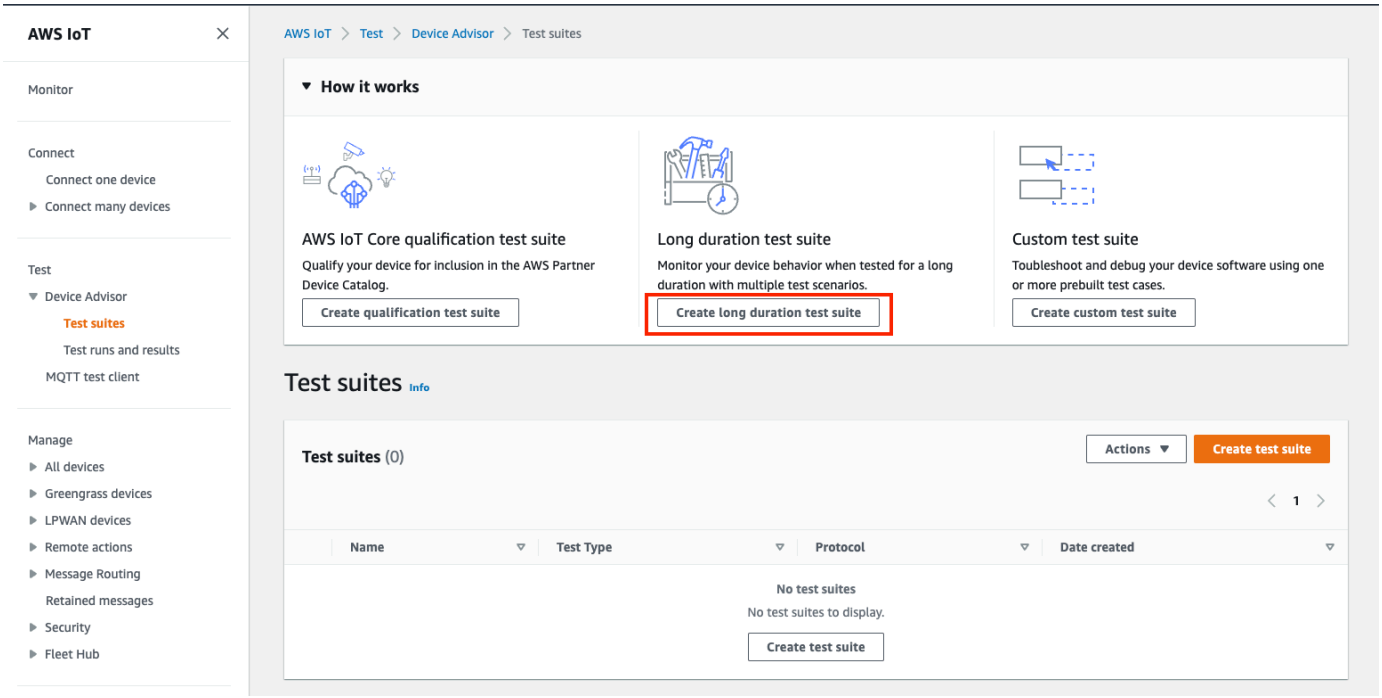
Si vous avez choisi l'option Utiliser la suite de tests de AWS IoT qualification lors de la création d'une suite de tests et que vous avez pu exécuter une suite de tests de qualification, vous pouvez télécharger un rapport de qualification en choisissant Télécharger le rapport de qualification sur la page de résumé des tests.



Flux de travail de la console de tests de longue durée

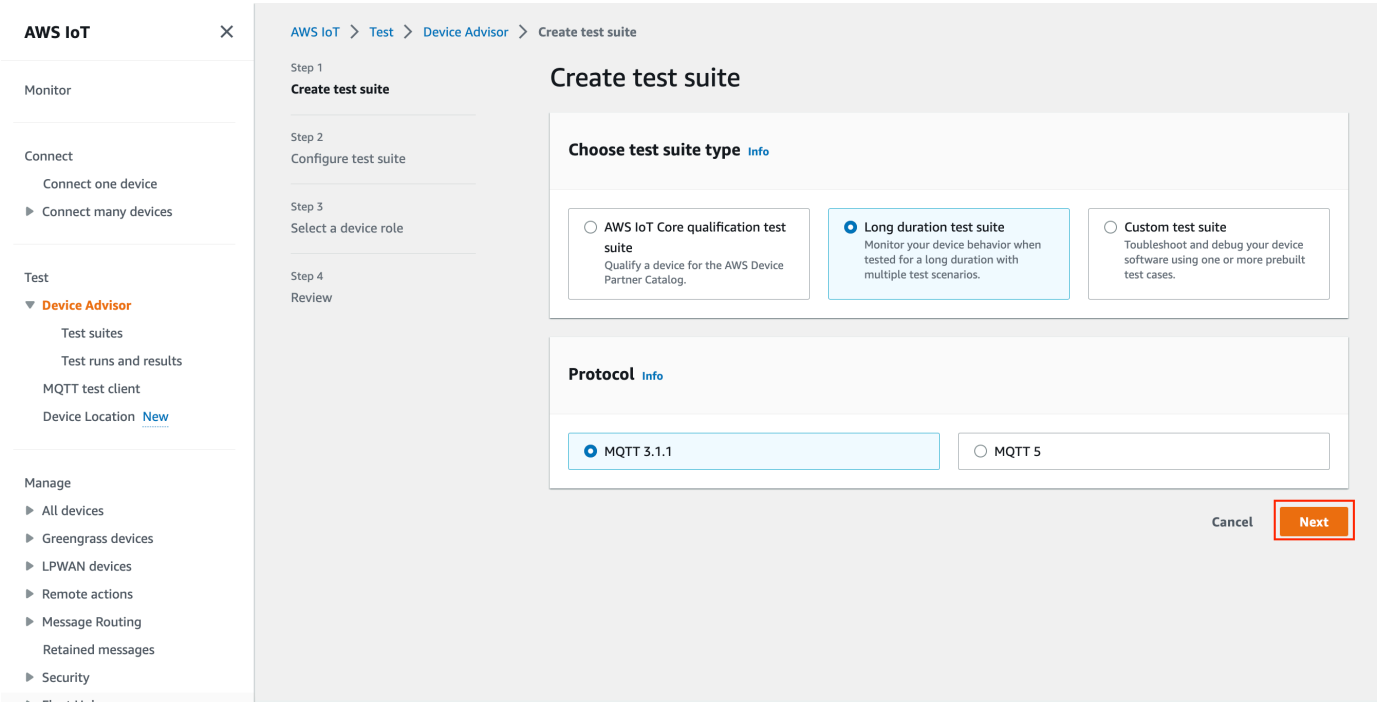
Ce didacticiel vous aide à démarrer les tests de longue durée sur Device Advisor à l'aide de la console. Pour terminer le didacticiel, suivez les étapes indiquées sur Configuration.

1. Dans la AWS IoT console, dans le volet de navigation, développez Test, Device Advisor, puis choisissez Suites de tests. Sur la page, sélectionnez Créer une suite de tests de longue durée.



2. Sur la page Créer une suite de tests, sélectionnez Suite de tests de longue durée, puis cliquez sur Suivant.

   Pour le protocole, choisissez MQTT 3.1.1 ou MQTT 5.



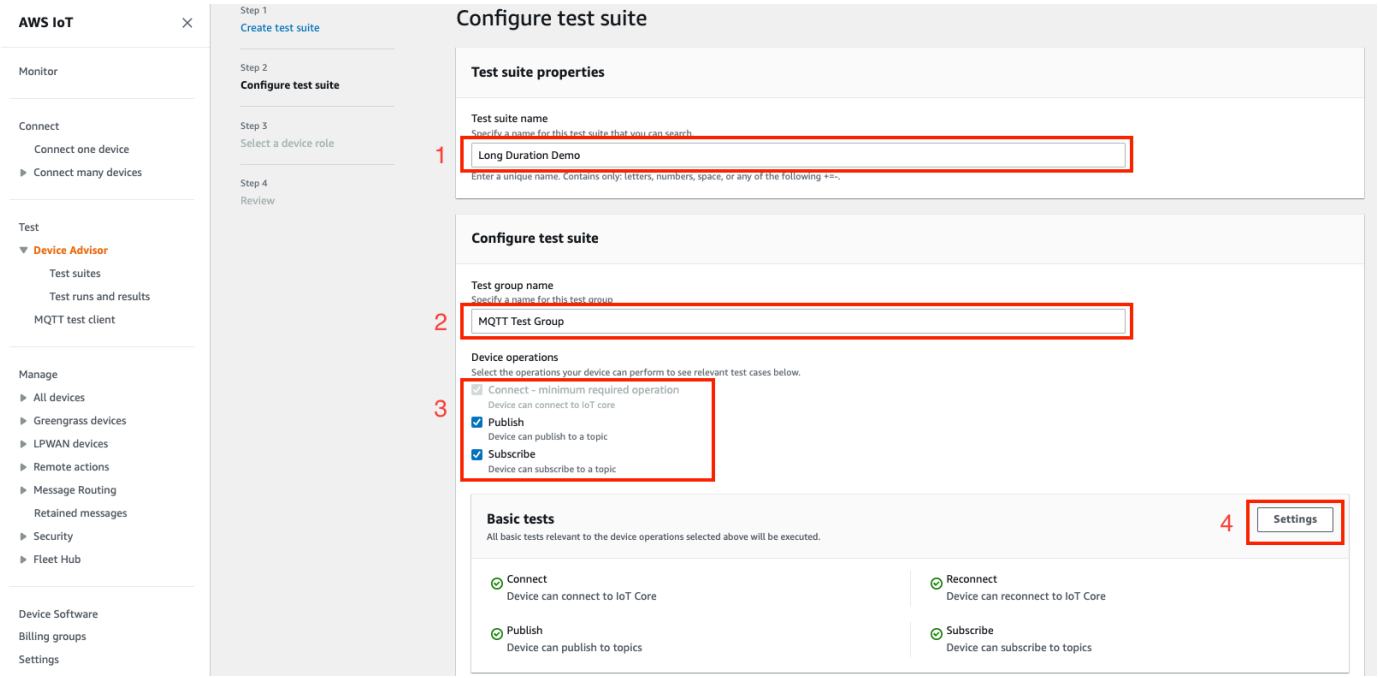
3. Sur la page Configurer l'événement de test, procédez de la façon suivante :

   a. Mettez à jour le champ Nom de la suite de tests.

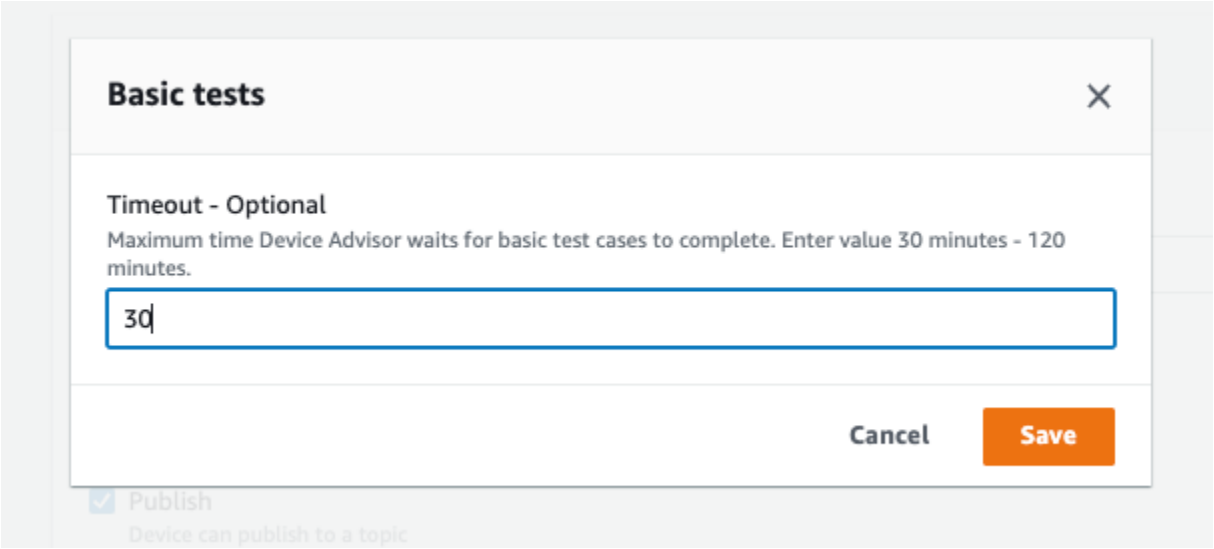
b. Mettez à jour le champ Nom de groupe de tests.

c. Choisissez les opérations que l'appareil peut effectuer. Cela permettra de sélectionner les tests à exécuter.

d. Sélectionnez l'option Paramètres.



4. (Facultatif) Entrez la durée maximale pendant laquelle Device Advisor doit attendre la fin des tests de base. Sélectionnez Save.



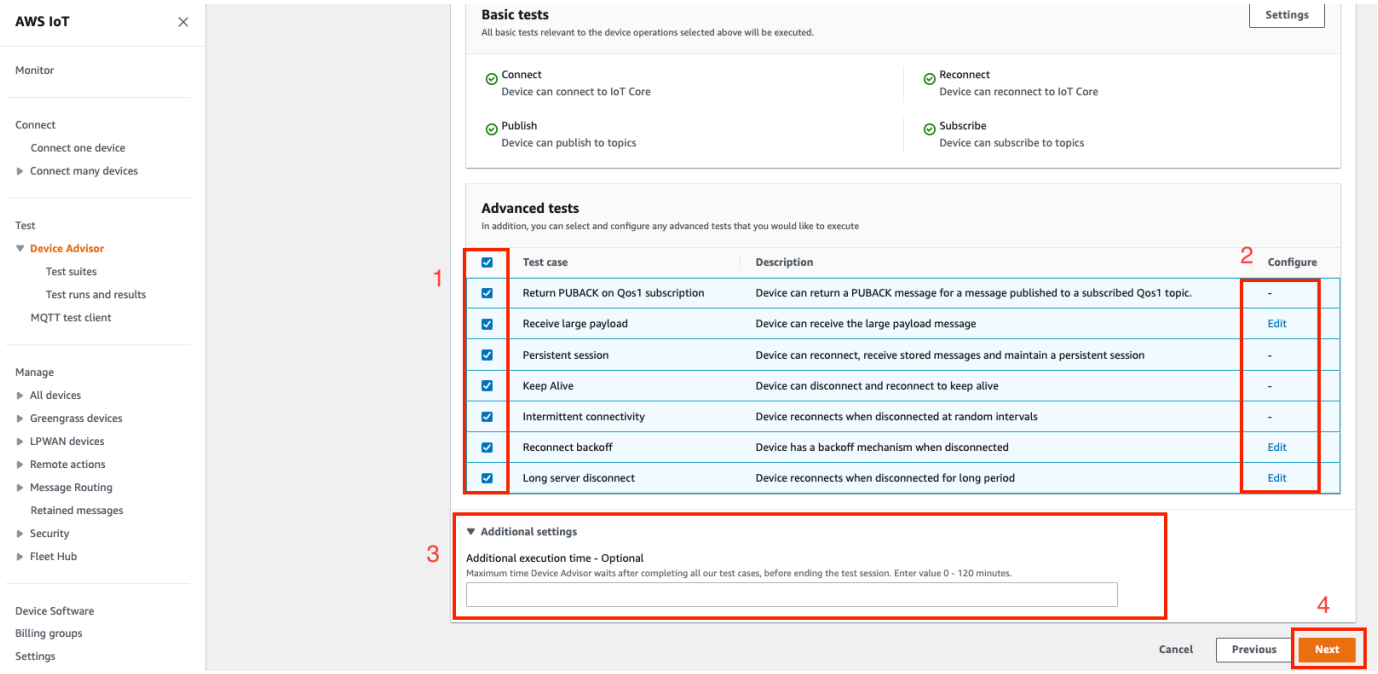
5. Procédez comme suit dans les sections Tests avancés et Paramètres supplémentaires.

a. Sélectionnez ou désélectionnez les tests avancés que vous souhaitez exécuter dans le cadre de ce test.

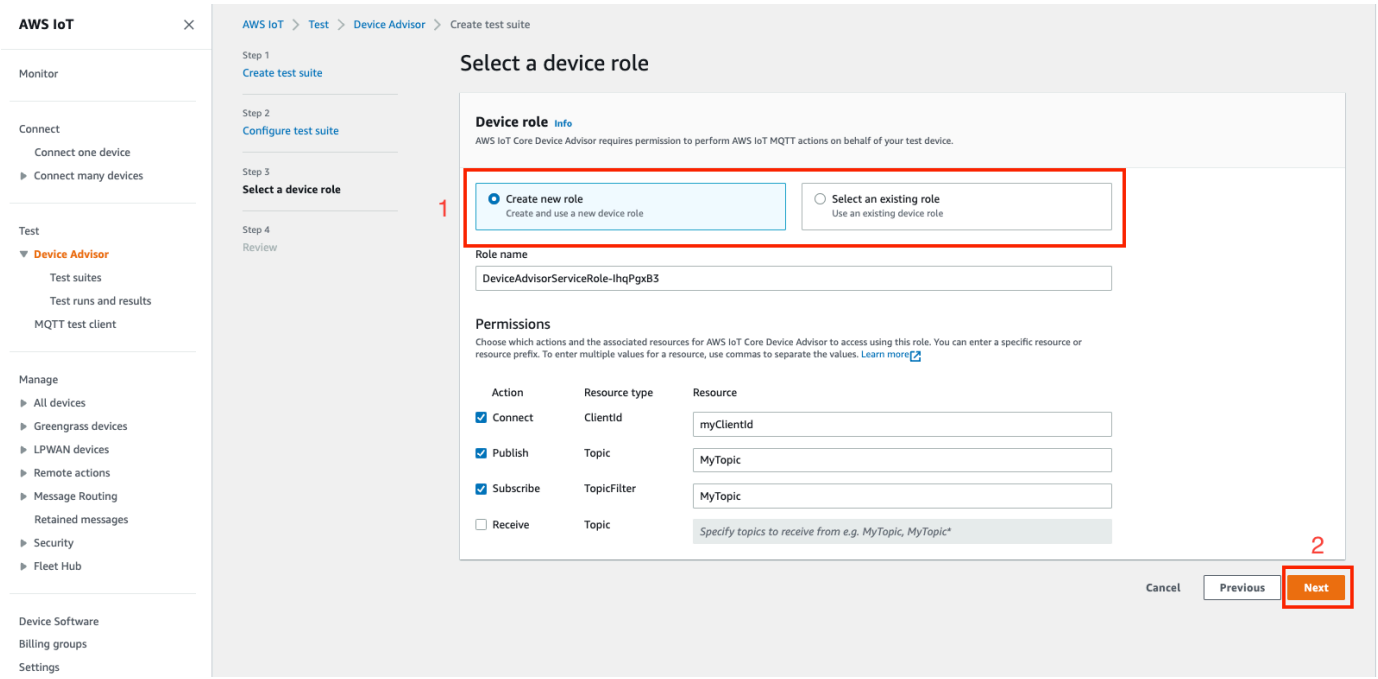
b. Modifiez les configurations pour les tests, le cas échéant.

c. Configurez le temps d'exécution supplémentaire dans la section Paramètres supplémentaires.

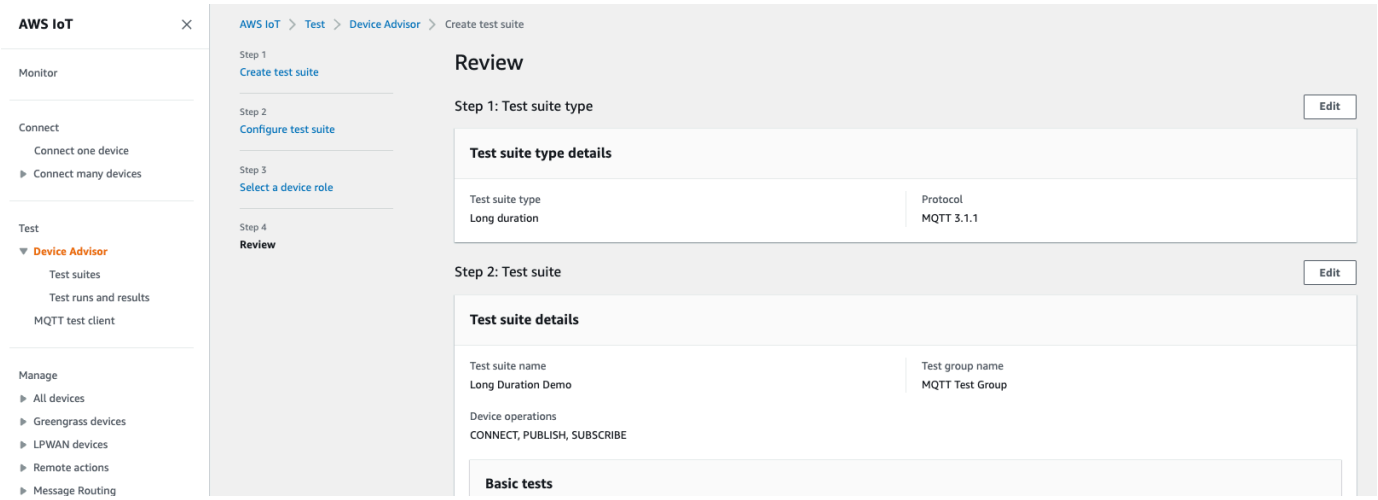
d. Choisissez Next (Suivant) pour passer à l'étape suivante.

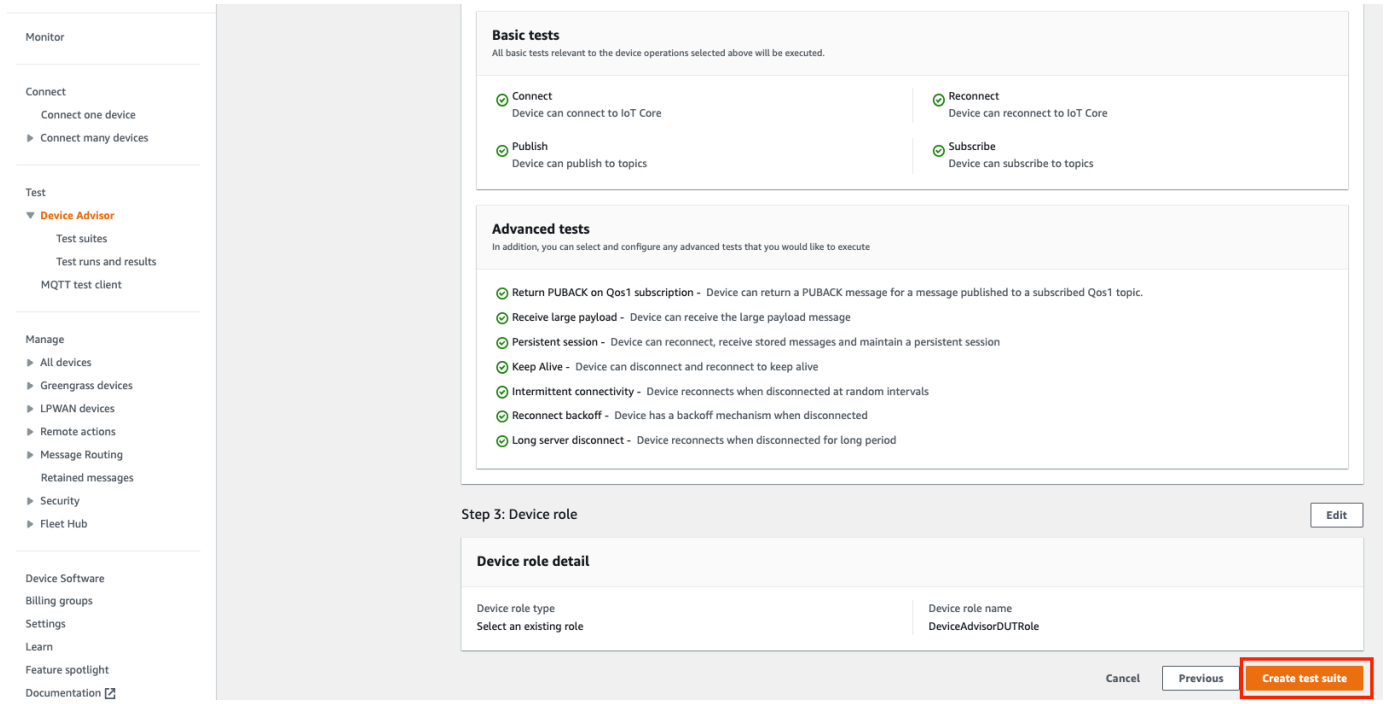


6. Dans cette étape, créez un nouveau rôle ou sélectionnez un rôle existant. Consultez Créer un rôle IAM à utiliser comme rôle de périphérique pour plus de détails.

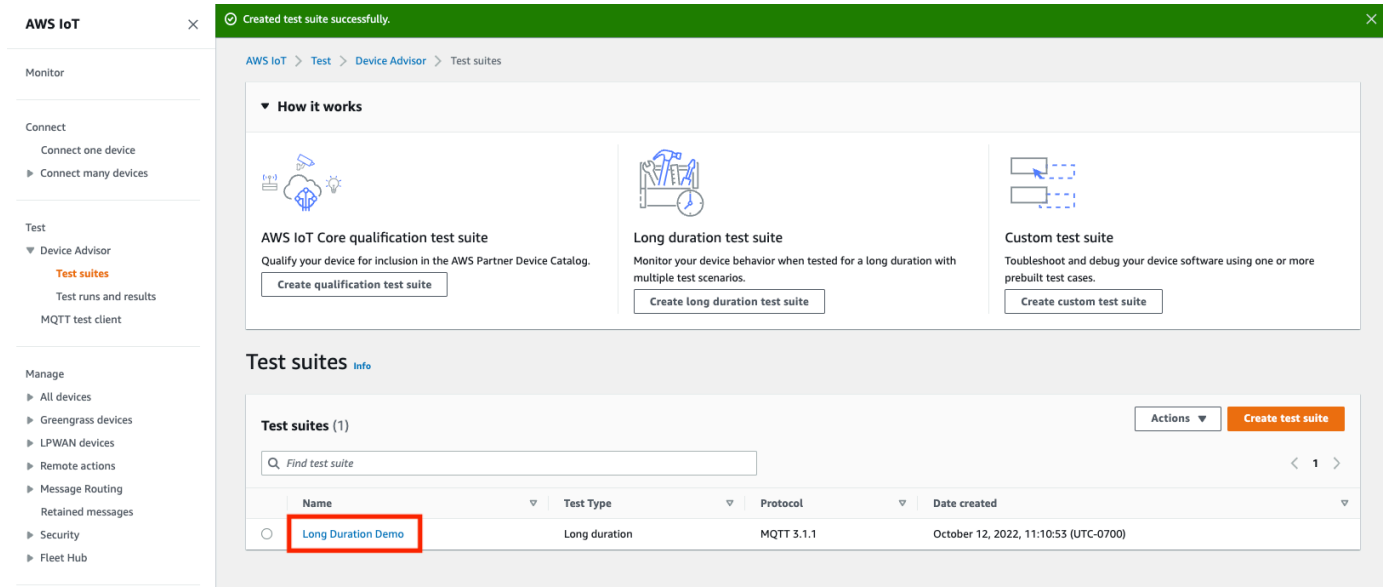


7. Passez en revue toutes les configurations créées jusqu'à cette étape et sélectionnez Créer une suite de tests.

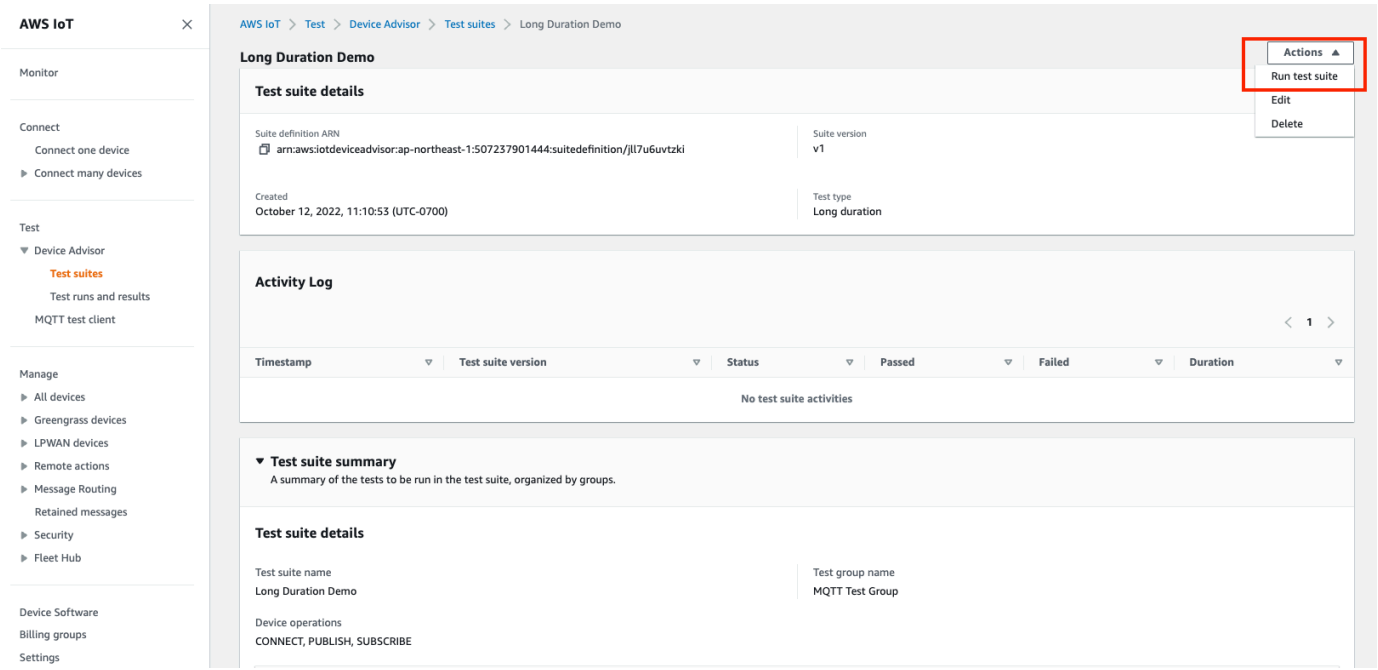




8. La suite de tests créée se trouve dans la section Suites de tests. Sélectionnez la suite pour afficher les détails.



9. Pour exécuter la suite de tests créée, sélectionnez Actions puis Exécuter la suite de tests.

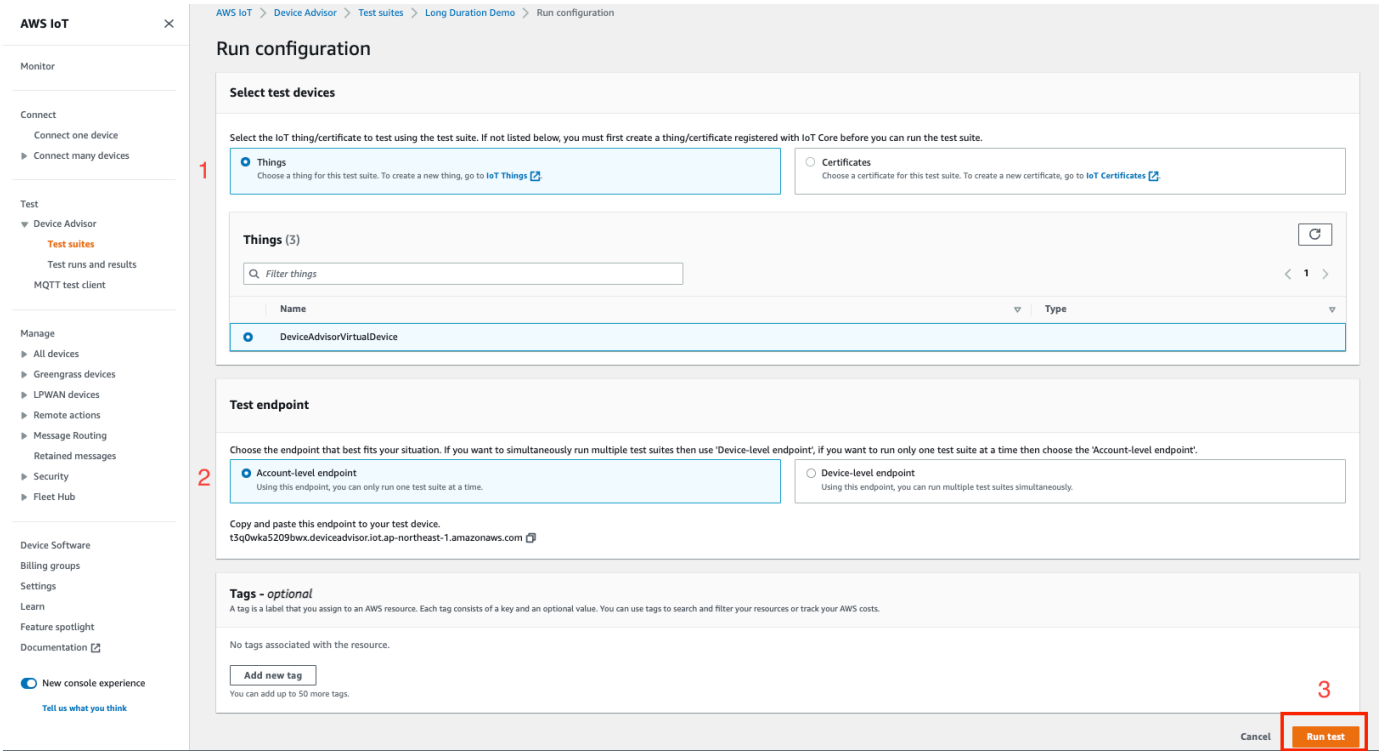


10. Choisissez les options de configuration sur la page Exécuter la configuration.

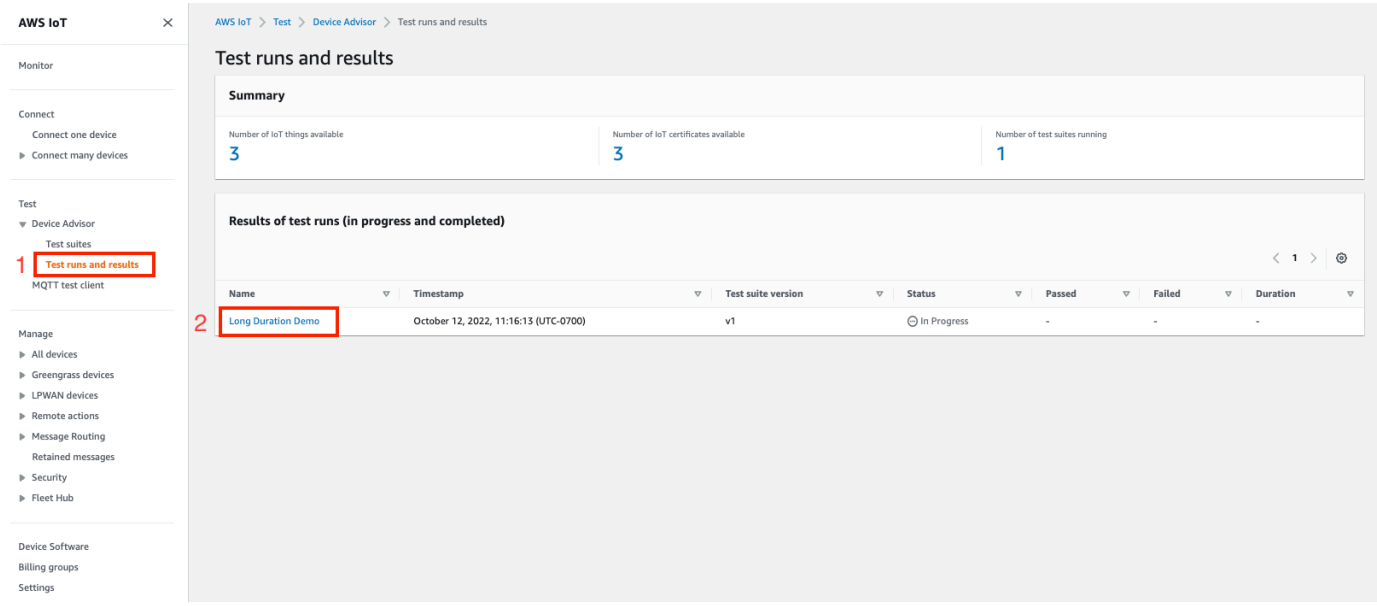
    a. Sélectionnez les objets ou le certificat sur lesquels exécuter le test.

    b. Sélectionnez le point de terminaison au niveau du compte ou le point de terminaison au niveau de l'appareil.

    c. Pour exécuter le test, choisissez Exécuter le test.

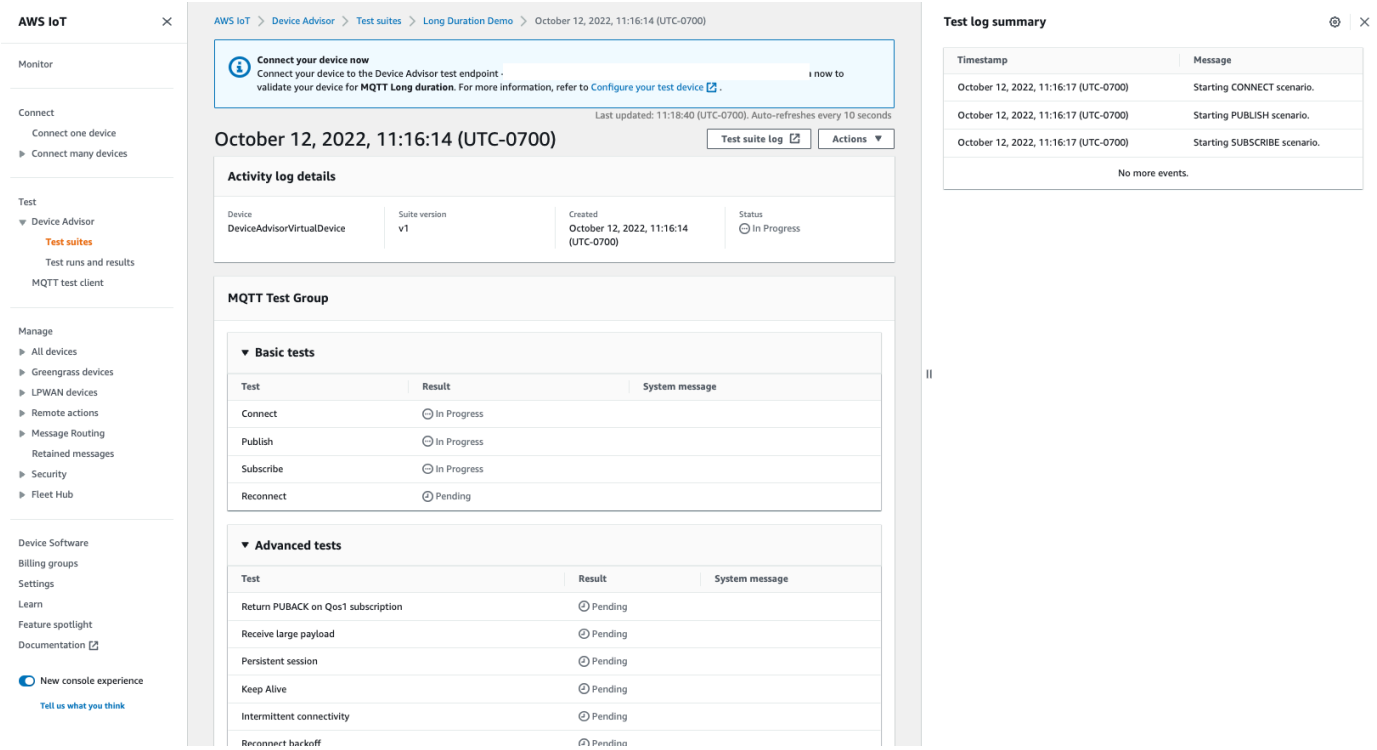


11. Pour afficher les résultats de l'exécution de la suite de tests, sélectionnez Exécutions de tests et résultats dans le volet de navigation de gauche. Choisissez la suite de tests exécutée pour afficher le détail des résultats.

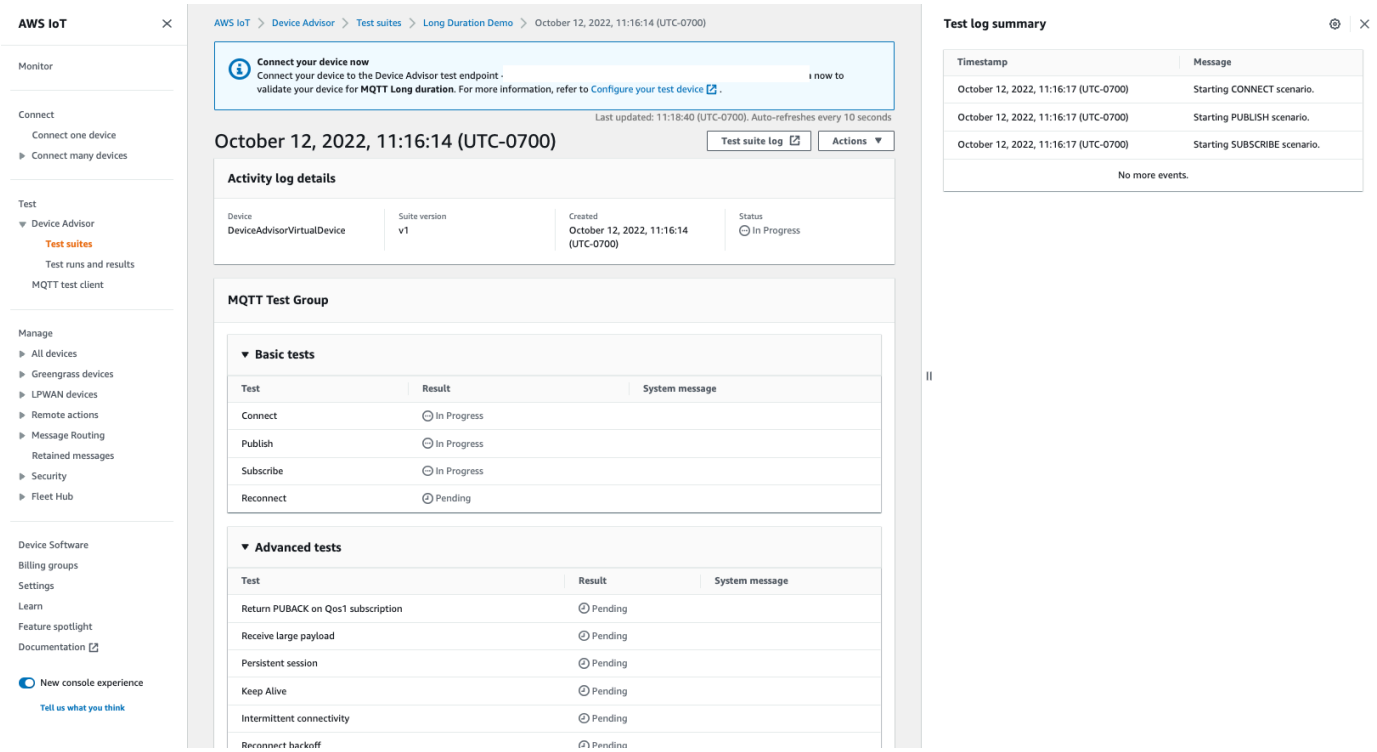


12. L'étape précédente fait apparaître la page de résumé du test. Tous les détails du test sont affichés sur cette page. Lorsque la console vous invite à démarrer la connexion de l'appareil,

connectez votre appareil au point de terminaison fourni. La progression des tests est visible sur cette page.



13. Le test de longue durée fournit un résumé supplémentaire du journal de test sur le panneau latéral qui affiche tous les événements importants survenus entre l'appareil et le courtier en temps quasi réel. Pour afficher des journaux détaillés, cliquez sur Journal des cas de test.



Points de terminaison d’un VPC Device Advisor (AWS PrivateLink)

Vous pouvez établir une connexion privée entre votre VPC et le point de terminaison de AWS IoT Core Device Advisor test (plan de données) en créant un point de terminaison VPC d'interface. Vous pouvez utiliser ce point de terminaison pour valider AWS IoT les appareils afin de garantir une connectivité fiable et sécurisée AWS IoT Core avant de les déployer en production. Pour accéder aux journaux CloudWatch pour l'exécution de la suite de tests, choisissez Journal de la suite de tests.

AWS PrivateLinkalimente les points de terminaison d'interface utilisés avec vos appareils IoT. Ce service vous aide à accéder en privé au point de terminaison de test AWS IoT Core Device Advisor sans passerelle Internet, périphérique NAT, connexion VPN ni connexion AWS Direct Connect . Les instances de votre VPC qui envoient des paquets TCP et MQTT n'ont pas besoin d'adresses IP publiques pour communiquer avec les points de terminaison de test. AWS IoT Core Device Advisor Le trafic entre votre VPC et celui qui AWS IoT Core Device Advisor ne sort pas. AWS Cloud Toutes les communications TLS et MQTT entre les appareils IoT et les scénarios de test Device Advisor restent dans les limites de vos ressources. Compte AWS

Chaque point de terminaison d'interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux.

Pour en savoir plus sur l'utilisation des points de terminaison d'un VPC d'interface, consultez Points de terminaison de VPC d'interface (AWS PrivateLink) dans le Guide de l'utilisateur Amazon VPC.

Considérations relatives aux points de AWS IoT Core Device Advisor terminaison VPC

Consultez les propriétés et les limitations du point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC avant de configurer les points de terminaison d'un VPC d'interface. Considérez les points suivants avant de continuer :

- AWS IoT Core Device Advisor prend actuellement en charge les appels vers le point de terminaison de test Device Advisor (plan de données) depuis votre VPC. Un agent de messages utilise les communications par plan de données pour envoyer et recevoir des données. Il le fait à l'aide de paquets TLS et MQTT. Points de terminaison VPC pour AWS IoT Core Device Advisor connecter votre AWS IoT appareil aux points de terminaison de test Device Advisor. Les actions d'API du plan de contrôle ne sont pas utilisées par ce point de terminaison VPC. Pour créer ou exécuter une suite de tests ou un autre plan de contrôle APIs, utilisez la console, un AWS SDK ou une interface de ligne de AWS commande sur Internet public.
- Les points de terminaison VPC suivants Régions AWS prennent en charge : AWS IoT Core Device Advisor
  - USA Est (Virginie du Nord)
  - USA Ouest (Oregon)
  - Asie Pacifique (Tokyo)
  - Europe (Irlande)
- Device Advisor prend en charge le MQTT avec les certificats client X.509 et les certificats de serveur RSA.
- Les politiques de point de terminaison d'un VPC ne sont pas prises en charge pour le moment.
- Consultez les conditions requises pour les points de terminaison VPC pour obtenir des instructions sur la façon de créer des ressources qui connectent les points de terminaison VPC. Vous devez créer un VPC et des sous-réseaux privés pour utiliser les points de terminaison AWS IoT Core Device Advisor VPC.
- Vos AWS PrivateLink ressources sont soumises à des quotas. Pour plus d'informations, consultez AWS PrivateLink Quotas.
- Les points de terminaison VPC ne prennent en charge que le trafic. IPv4

Créer un point de terminaison de VPC d'interface pour AWS IoT Core Device Advisor

Pour démarrer avec les points de terminaison d'un VPC, créez un point de terminaison d'un VPC d'interface. Ensuite, sélectionnez AWS IoT Core Device Advisor comme Service AWS. Si vous utilisez le AWS CLI, appelez describe-vpc-endpoint-servicespour confirmer qu'il AWS IoT Core Device Advisor est présent dans une zone de disponibilité de votre Région AWS. Vérifiez que le groupe de sécurité attaché au point de terminaison autorise les communications par protocole TCP pour le trafic MQTT et TLS. Par exemple, dans la région USA Est (Virginie du Nord), utilisez la commande suivante :

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.deviceadvisor.iot
```

Vous pouvez créer un point de terminaison VPC à l' AWS IoT Core aide du nom de service suivant :

- com.amazonaws.region.deviceadvisor.iot

Par défaut, le DNS privé est activé pour le point de terminaison. Cela garantit que l'utilisation du point de terminaison de test par défaut reste dans vos sous-réseaux privés. Pour obtenir un point de terminaison au niveau de votre compte ou de votre appareil, utilisez la console AWS CLI ou un AWS SDK. Par exemple, si vous exécutez get-endpoint dans un sous-réseau public ou sur Internet public, vous pouvez obtenir votre point de terminaison et l'utiliser pour vous connecter à Device Advisor. Pour plus d'informations, consultez Accès à un service via un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Pour connecter les clients MQTT aux interfaces de point de terminaison du VPC, AWS PrivateLink le service crée des enregistrements DNS dans une zone hébergée privée attachée à votre VPC. Ces enregistrements DNS dirigent les demandes de l'appareil AWS IoT vers le point de terminaison du VPC.

Contrôle de l'accès aux points de AWS IoT Core Device Advisor terminaison via VPC

Vous pouvez restreindre l'accès des appareils aux points de terminaison VPC AWS IoT Core Device Advisor et autoriser l'accès uniquement via ces points en utilisant les clés contextuelles de condition VPC. AWS IoT Core prend en charge les clés de contexte liées au VPC suivantes :

- SourceVpc
- SourceVpce
- VPCSourceIp

> **Note**
>
> AWS IoT Core Device Advisor ne prend pas en charge les politiques de point de terminaison VPC pour le moment.

La politique suivante autorise la connexion à AWS IoT Core Device Advisor l'aide d'un ID client correspondant au nom de l'objet. Il publie également sur n'importe quelle rubrique préfixée par le nom de l'objet. La politique dépend de la connexion de l'appareil à un point de terminaison d'un VPC avec un ID de point de terminaison d'un VPC particulier. Cette politique refuse les tentatives de connexion à votre point de terminaison de test AWS IoT Core Device Advisor public.

JSON

```
{
"Version": "2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
"StringEquals": {
"aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }

        },
        {
"Effect": "Allow",
```

```
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Cas de test Device Advisor

Device Advisor propose des tests prédéfinis dans six catégories.

- TLS
- MQTT
- Shadow
- Exécution d'une tâche
- Autorisations et politiques
- Tests de longue durée

Device Advisor teste des scénarios pour se qualifier pour le programme de qualification des AWS appareils.

Votre appareil doit réussir les tests suivants pour être éligible conformément au AWS Device Qualification Program. (Programme de qualification des appareils)

> **Note**
>
> Il s'agit d'une liste révisée des tests de qualification.

- Connexion TLS (« Connexion TLS »)

- Certificat de serveur de nom de sujet incorrect TLS (« Nom commun du sujet (CN) /nom alternatif du sujet (SAN) incorrect »)
- Certificat de serveur TLS non sécurisé (« Non signé par une autorité de certification reconnue »)
- Support des appareils TLS pour les suites de AWS IoT chiffrement (« Support des appareils TLS pour les suites de chiffrement AWS IoT recommandées »)
- TLS Receive Maximum Size Fragments (« TLS reçoit des fragments de taille maximale »)
- TLS Expired Server Cert (« Certificat de serveur expiré »)
- TLS Large Size Server Cert (« certificat de serveur TLS de grande taille »)
- MQTT Connect (« L'appareil envoie le CONNECT à AWS IoT Core (Happy case) »)
- S'abonner à MQTT (« Peut s'abonner (Happy Case) »)
- MQTT Publish (« QoS0 (Happy Case) »)
- MQTT Connect Jitter Retries (« Nouvelles tentatives de connexion de l'appareil avec intervalle de gigue - Aucune réponse CONNACK »)

TLS

Utilisez ces tests pour déterminer si le protocole de sécurité de la couche transport (TLS) entre vos appareils AWS IoT est sécurisé.

> **Note**
>
> Device Advisor prend désormais en charge TLS 1.3.

Happy Path

Connexion TLS

Valide si le périphérique testé peut effectuer la prise de contact TLS avec. AWS IoT Ce test ne valide pas l'implémentation MQTT de l'appareil client.

Example Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Pour de meilleurs résultats, nous recommandons un délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_tls_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Connect",
         "version":"0.0.0"
      }
   }
]
```

Example Sorties du scénario de test :

- Réussite — L'appareil testé a terminé la prise de contact TLS avec. AWS IoT
- Réussir avec des avertissements — L'appareil testé a terminé la prise de contact TLS avec AWS IoT, mais des messages d'avertissement TLS ont été envoyés par l'appareil ou. AWS IoT
- Échec : le périphérique testé n'a pas réussi à terminer la prise de contact TLS AWS IoT en raison d'une erreur de connexion.

TLS reçoit des fragments de taille maximale

Ce cas de test confirme que votre appareil peut recevoir et traiter des fragments de taille maximale TLS. Votre appareil de test doit s'abonner à une rubrique préconfigurée avec QoS 1 pour recevoir une charge utile importante. Vous pouvez personnaliser la charge utile avec la configuration `${payload}`.

Example Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Pour de meilleurs résultats, nous recommandons un délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"TLS Receive Maximum Size Fragments",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
         "PAYLOAD_FORMAT":"{"message":"${payload}"}", // A string with a placeholder
 ${payload}, or leave it empty to receive a plain string.
         "TRIGGER_TOPIC": "test_1" // A topic to which a device will subscribe, and
 to which a test case will publish a large payload.
      },
      "test":{
         "id":"TLS_Receive_Maximum_Size_Fragments",
         "version":"0.0.0"
      }
   }
]
```

Suites de chiffrement

Support des appareils TLS pour les suites de AWS IoT chiffrement recommandées

Vérifie que les suites de chiffrement figurant dans le message Hello du client TLS envoyé par le périphérique testé contiennent les AWS IoT cipher suites.(suites de chiffrement recommandées) Il fournit des informations supplémentaires sur les suites de chiffrement prises en charge par l'appareil.

Example Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_tls_support_aws_iot_cipher_suites_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Support_AWS_IoT_Cipher_Suites",
         "version":"0.0.0"
      }
   }
]
```

Example Sorties du scénario de test :

- Réussite — L'appareil testé contient au moins l'une des suites de AWS IoT chiffrement recommandées et ne contient aucune suite de chiffrement non prise en charge.
- Passez avec avertissements : les suites de chiffrement de l'appareil contiennent au moins une suite de chiffrement AWS IoT , mais :

  1. Il ne contient aucune des suites de chiffrement recommandées
  2. Il contient des suites de chiffrement qui ne sont pas prises en charge par AWS IoT.

  Nous vous suggérons de vérifier que toutes les suites de chiffrement non prises en charge sont sûres.
- Échec : le périphérique soumis aux suites de chiffrement testées ne contient aucune des suites de chiffrement AWS IoT prises en charge.

Certificat de serveur de plus grande taille

Certificat de serveur TLS de grande taille

Les validations sur votre appareil peuvent terminer la négociation TLS avec AWS IoT lorsqu'il reçoit et traite un certificat de serveur de plus grande taille. La taille du certificat de serveur (en octets) utilisé par ce test est supérieure de 20 à celle actuellement utilisée dans le cas de test TLS Connect et IoT Core. Au cours de ce scénario de AWS IoT test, testez l'espace tampon de votre appareil pour le TLS. Si l'espace tampon est suffisamment important, la poignée de main TLS se termine sans erreur. Ce test ne valide pas l'implémentation MQTT de l'appareil. Le scénario de test prend fin une fois le processus de handshake TLS terminé.

Example Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Pour de meilleurs résultats, nous recommandons un délai d'attente de 2 minutes. Si ce scénario de test échoue mais que le test TLS Connect réussit, nous vous recommandons d'augmenter la limite d'espace tampon de votre appareil pour le protocole TLS. L'augmentation de la limite d'espace tampon garantit que votre appareil pourra traiter un certificat de serveur de plus grande taille au cas où la taille augmenterait.

```
"tests":[
   {
      "name":"my_tls_large_size_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Large_Size_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Sorties du scénario de test :

- Réussite — L'appareil testé a terminé la prise de contact TLS avec. AWS IoT

- Réussissez avec des avertissements — L'appareil testé a terminé la prise de contact TLS avec AWS IoT, mais des messages d'avertissement TLS proviennent soit de l'appareil, soit. AWS IoT
- Échec : le périphérique testé n'a pas réussi à terminer la prise de contact TLS en AWS IoT raison d'une erreur survenue lors du processus de prise de contact.

Certificat de serveur non sécurisé TLS

Non signé par une autorité de certification reconnue

Confirme que l'appareil testé ferme la connexion s'il est présenté avec un certificat de serveur sans signature valide de l'autorité de certification ATS. Un appareil ne doit se connecter qu'à un point de terminaison présentant un certificat valide.

Example Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_tls_unsecure_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Unsecure_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Sorties du scénario de test :

- Réussite — L'appareil testé a fermé la connexion.
- Échec — L'appareil testé a terminé la prise de contact TLS avec. AWS IoT

Certificat de serveur du nom de sujet incorrect TLS / Nom commun de sujet (CN) / Nom alternatif du sujet (SAN) incorrect

Valide que l'appareil testé ferme la connexion s'il reçoit un certificat de serveur pour un nom de domaine différent de celui demandé.

Example Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_tls_incorrect_subject_name_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"TLS_Incorrect_Subject_Name_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Sorties du scénario de test :

- Réussite — L'appareil testé a fermé la connexion.
- Échec : le périphérique testé a terminé la prise de contact TLS avec. AWS IoT

Certificat de serveur TLS expiré

Certificat de serveur expiré

Confirme que l'appareil testé ferme la connexion s'il reçoit un certificat de serveur expiré.

Example Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_tls_expired_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Expired_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example Sorties du scénario de test :

- Réussite — L'appareil testé refuse de terminer la prise de contact TLS avec. AWS IoT L'appareil envoie un message d'alerte TLS avant de fermer la connexion.
- Pass with warnings — L'appareil testé refuse de terminer la prise de contact TLS avec AWS IoT. Cependant, il n'envoie pas de message d'alerte TLS avant de fermer la connexion.
- Échec : le périphérique testé termine la prise de contact TLS avec. AWS IoT

MQTT

CONNECTEZ, DÉCONNECTEZ et RECONNECTEZ

« L'appareil envoie le CONNECT à AWS IoT Core (Happy case) »

Valide que le périphérique testé envoie une demande CONNECT.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_mqtt_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"MQTT_Connect",
         "version":"0.0.0"
      }
   }
]
```

« L'appareil peut renvoyer PUBACK à un sujet arbitraire pour QoS1 »

Ce cas de test vérifiera si l'appareil (client) peut renvoyer un message PUBACK s'il a reçu un message de publication du broker après s'être abonné à un sujet avec QoS1.

Le contenu et la taille de la charge utile sont configurables pour ce cas de test. Si la taille de la charge utile est configurée, Device Advisor écrasera la valeur du contenu de la charge utile et enverra une charge utile prédéfinie au périphérique avec la taille souhaitée. La taille de la charge utile est une valeur comprise entre 0 et 128 et ne peut pas dépasser 128 Ko. AWS IoT Core rejette les demandes de publication et de connexion supérieures à 128 Ko, comme indiqué sur la page AWS IoT Core agent de messages et des limites et quotas du protocole.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes. PAYLOAD_SIZE peut être configuré à une valeur comprise entre 0 et 128 kilo-octets. La définition d'une taille de charge utile remplace le

contenu de la charge utile, car Device Advisor renverra une charge utile prédéfinie avec la taille donnée à l'appareil.

```
"tests":[
{
        "name":"my_mqtt_client_puback_qos1",
        "configuration": {
            // optional:"TRIGGER_TOPIC": "myTopic",
            "EXECUTION_TIMEOUT":"300", // in seconds
            "PAYLOAD_FOR_PUBLISH_VALIDATION":"custom payload",
            "PAYLOAD_SIZE":"100" // in kilobytes
        },
        "test": {
            "id": "MQTT_Client_Puback_QoS1",
            "version": "0.0.0"
        }
    }
]
```

« Device connect réessaie avec intervalle de gigue - Aucune réponse CONNACK"

Vérifie que le périphérique testé utilise le système de réduction de gigue approprié lorsqu'il se reconnecte au courtier au moins cinq fois. Le courtier enregistre l'horodatage de la demande CONNECT de l'appareil testé, effectue la validation des paquets, fait une pause sans envoyer de CONNACK à l'appareil testé et attend que l'appareil testé renvoie la demande. La sixième tentative de connexion est autorisée à passer et CONNACK est autorisé à revenir vers l'appareil testé.

Le processus précédent est recommencé. Au total, ce cas de test nécessite que l'appareil se connecte au moins 12 fois. Les horodatages collectés sont utilisés pour valider que l'atténuation de la gigue est utilisée par l'appareil testé. Si le délai de temporisation de l'appareil testé est strictement exponentiel, ce scénario de test sera validé avec des avertissements.

Nous recommandons d'implémenter le mécanisme Backoff exponentiel et Gigue sur l'appareil testé pour réussir ce scénario de test.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 4 minutes.

```
"tests":[
   {
      "name":"my_mqtt_jitter_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",    // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Jitter_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

« Device connect réessaie avec backoff exponentiel - Aucune réponse CONNACK"

Vérifie que l'appareil testé utilise le backoff exponentiel approprié lors de la reconnexion au courtier au moins cinq fois. Le courtier enregistre l'horodatage de la requête CONNECT de l'appareil testé, effectue la validation des paquets, fait une pause sans envoyer de CONNACK à l'appareil client et attend que l'appareil testé renvoie la demande. Les horodatages collectés sont utilisés pour valider qu'une backoff exponentiel est utilisée par l'appareil testé.

Nous recommandons d'implémenter le mécanisme Backoff exponentiel et Gigue sur l'appareil testé pour réussir ce scénario de test.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 4 minutes.

```
"tests":[
```

```
   {
      "name":"my_mqtt_exponential_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"600",  // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Exponential_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

« Reconnexion de l'appareil avec atténuation de la gigue - Après la déconnexion du serveur »

Valide si un appareil testé utilise l'instabilité et le ralentissement nécessaire lors de la reconnexion après avoir été déconnecté du serveur. Device Advisor déconnecte l'appareil du serveur au moins cinq fois et observe le comportement de l'appareil lors de la reconnexion MQTT. Device Advisor enregistre l'horodatage de la demande CONNECT pour le périphérique testé, effectue la validation des paquets, fait une pause sans envoyer de CONNACK à l'appareil client et attend que l'appareil testé renvoie la demande. Les horodatages collectés sont utilisés pour valider que l'appareil testé utilise la gigue et l'interruption lors de la reconnexion. Si l'appareil testé présente une backoff exponentiel stricte ou n'implémente pas un mécanisme d'atténuation de gigue approprié, ce scénario de test réussira avec des avertissements. Si le dispositif testé a mis en œuvre un mécanisme d'arrêt linéaire ou constant, le test échouera.

Pour réussir ce cas de test, nous vous recommandons d'implémenter Backoff exponentiel et Gigue sur l'appareil testé dans ce test.

Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 4 minutes.
> Le nombre de tentatives de reconnexion pour valider le backoff peut être modifié en spécifiant le `RECONNECTION_ATTEMPTS`. Le nombre doit être compris entre 5 et 10. La valeur par défaut est 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_server_disconnect",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Server_Disconnect",
         "version":"0.0.0"
      }
   }
]
```

« Reconnexion de l'appareil avec réduction de la gigue - Sur connexion instable »

Valide si un appareil testé utilise la gigue et l'intervalle de temps nécessaires lors de la reconnexion sur une connexion instable. Device Advisor déconnecte l'appareil du serveur après cinq connexions réussies et observe le comportement de l'appareil pour la reconnexion MQTT. Device Advisor enregistre l'horodatage de la demande CONNECT pour l'appareil testé, effectue la validation des paquets, renvoie CONNACK, se déconnecte, enregistre l'horodatage de la déconnexion et attend que l'appareil testé renvoie la demande. Les horodatages collectés sont utilisés pour valider que l'appareil testé utilise la gigue et l'interruption lors de la reconnexion après des connexions réussies mais instables. Si l'appareil testé présente une backoff exponentiel stricte ou n'implémente pas un mécanisme d'atténuation de gigue approprié, ce scénario de test réussira avec des avertissements. Si le dispositif testé a mis en œuvre un mécanisme d'arrêt linéaire ou constant, le test échouera.

Pour réussir ce cas de test, nous vous recommandons d'implémenter Backoff exponentiel et Gigue sur l'appareil testé dans ce test.

Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 4 minutes.

Le nombre de tentatives de reconnexion pour valider le backoff peut être modifié en spécifiant le `RECONNECTION_ATTEMPTS`. Le nombre doit être compris entre 5 et 10. La valeur par défaut est 5.

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_unstable_connection",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Unstable_Connection",
         "version":"0.0.0"
      }
   }
]
```

Publication

« QoS0 (Happy Case) »

Valide que l'appareil testé publie un message avec QoS0 ou QoS1. Vous pouvez également valider la rubrique du message et la charge utile en spécifiant la valeur de la rubrique et la charge utile dans les paramètres de test.

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_mqtt_publish_test",
      "configuration":{
         // optional:
```

```
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish",
         "version":"0.0.0"
      }
   }
]
```

« Nouvelle tentative de publication de QoS1 - Pas de PUBACK »

Valide que l'appareil testé republie un message envoyé avec QoS1, si le courtier n'envoie pas PUBACK. Vous pouvez également valider le sujet du message en précisant ce sujet dans les paramètres du test. L'appareil client ne doit pas se déconnecter avant de republier le message. Ce test permet également de vérifier que le message republié possède le même identifiant de paquet que l'original. Pendant l'exécution du test, si l'appareil perd la connexion et se reconnecte, le scénario de test sera réinitialisé sans échec et l'appareil doit recommencer les étapes du scénario de test.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Il est recommandé de le faire pendant au moins 4 minutes.

```
"tests":[
   {
      "name":"my_mqtt_publish_retry_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retry_No_Puback",
         "version":"0.0.0"
```

```
      }
   }
]
```

« Publier les messages conservés »

Valide que l'appareil testé publie un message `retainFlag` set to true. (défini sur true) Vous pouvez valider la rubrique et la charge utile du message en définissant la valeur de rubrique et la charge utile dans les paramètres de test. Si le paramètre `retainFlag` envoyé dans le paquet PUBLISH n'est pas défini sur true, le scénario de test échouera.

Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes. Pour exécuter ce scénario de test, ajoutez l'action`iot:RetainPublish` dans rôle de votre appareil.

```
"tests":[
   {
      "name":"my_mqtt_publish_retained_messages_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds

 "TOPIC_FOR_PUBLISH_RETAINED_VALIDATION": "my_TOPIC_FOR_PUBLISH_RETAINED_VALIDATION",

 "PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retained_Messages",
         "version":"0.0.0"
      }
   }
]
```

« Publier avec la propriété utilisateur »

Valide que l'appareil testé publie un message avec la propriété utilisateur correcte. Vous pouvez valider la propriété utilisateur en définissant la paire nom-valeur dans les paramètres de test. Si la propriété utilisateur n'est pas fournie ou ne correspond pas, le scénario de test échoue.

Définition du cas de test de l'API :

> **Note**
>
> Il s'agit d'un MQTT5 seul cas de test.
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_mqtt_user_property_test",
      "test":{
        "USER_PROPERTIES": [
            {"name": "name1", "value":"value1"},
            {"name": "name2", "value":"value2"}
        ],
        "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"MQTT_Publish_User_Property",
         "version":"0.0.0"
      }
   }
]
```

S'abonner

« Je peux m'abonner (Happy Case) »

Vérifie que l'appareil testé est abonné aux rubriques MQTT. Vous pouvez également valider la rubrique à laquelle l'appareil testé est abonné en spécifiant cette rubrique dans les paramètres de test.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_mqtt_subscribe_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["my_TOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe",
         "version":"0.0.0"
      }
   }
]
```

« Réessayer de s'abonner - Pas de SUBACK »

Confirme que l'appareil testé tente à nouveau un abonnement ayant échoué aux sujets MQTT. Le serveur attend alors et n'envoie pas de SUBACK. Si l'appareil client ne réessaye pas l'abonnement, le test échoue. L'appareil client doit réessayer l'abonnement qui a échoué avec le même identifiant de paquet. Vous pouvez également valider la rubrique à laquelle l'appareil testé est abonné en spécifiant cette rubrique dans les paramètres de test. Pendant l'exécution du test, si l'appareil perd la connexion et se reconnecte, le scénario de test sera réinitialisé sans échec et l'appareil doit recommencer les étapes du scénario de test.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 4 minutes.

```
"tests":[
   {
      "name":"my_mqtt_subscribe_retry_test",
      "configuration":{
         "EXECUTION_TIMEOUT":"300",  // in seconds
         // optional:
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
["myTOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe_Retry_No_Suback",
         "version":"0.0.0"
      }
   }
]
```

Keep-Alive

« Matt No Ak PingResp »

Ce cas de test valide si le périphérique testé se déconnecte lorsqu'il ne reçoit pas de réponse ping. Dans le cadre de ce scénario de test, Device Advisor bloque les réponses envoyées AWS IoT Core depuis les demandes de publication, d'abonnement et de ping. Il vérifie également si l'appareil testé déconnecte la connexion MQTT.

Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons un délai d'attente supérieur à 1,5 fois la valeur `keepAliveTime` .
> La durée maximale `keepAliveTime` ne doit pas dépasser 230 secondes pour ce test.

```
"tests":[
    {
       "name":"Mqtt No Ack PingResp",
       "configuration":
          //optional:
          "EXECUTION_TIMEOUT":"306",   // in seconds
```

```
        },
        "test":{
           "id":"MQTT_No_Ack_PingResp",
           "version":"0.0.0"
        }
    }
]
```

Session persistante

« Session persistante (Happy Case) »

Ce cas de test valide le comportement de l'appareil lorsqu'il est déconnecté d'une session persistante. Le scénario de test vérifie si l'appareil peut se reconnecter, reprendre les abonnements à ses rubriques de déclenchement sans se réabonner explicitement, recevoir les messages stockés dans les rubriques et fonctionner comme prévu pendant une session persistante. Lorsque ce scénario de test est réussi, cela indique que le dispositif client est capable de maintenir une session persistante avec le AWS IoT Core courtier de la manière attendue. Pour plus d'informations sur les sessions AWS IoT persistantes, consultez la section Utilisation des sessions persistantes MQTT.

Dans ce scénario de test, l'appareil client doit se CONNECTER au AWS IoT Core avec un indicateur de session propre défini sur false, puis s'abonner à une rubrique de déclenchement. Après un abonnement réussi, l'appareil sera déconnecté par AWS IoT Core Device Advisor. Lorsque l'appareil est déconnecté, une charge utile de message QoS 1 sera stockée dans cette rubrique. Device Advisor autorisera ensuite l'appareil client à se reconnecter au point de terminaison de test. À ce stade, étant donné qu'il existe une session persistante, le périphérique client est censé reprendre ses abonnements aux rubriques sans envoyer de paquets SUBSCRIBE supplémentaires et recevoir le message QoS 1 du courtier. Après la reconnexion, si le dispositif client se réabonne à nouveau à son sujet déclencheur en envoyant un paquet SUBSCRIBE supplémentaire and/or si le client ne reçoit pas le message stocké du sujet déclencheur, le scénario de test échouera.

Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'au moins 4 minutes. Lors de la première connexion, l'appareil client doit

s'abonner explicitement à un `TRIGGER_TOPIC` qui n'était pas abonné auparavant. Pour réussir le scénario de test, l'appareil client doit s'abonner avec succès à `TRIGGER_TOPIC` avec une QoS 1. Après la reconnexion, le dispositif client est censé comprendre qu'une session permanente est active ; il doit donc accepter le message stocké envoyé par la rubrique déclencheur et renvoyer PUBACK pour ce message spécifique.

```
"tests":[
   {
      "name":"my_mqtt_persistent_session_happy_case",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         // if Payload not provided, a string will be stored in the trigger topic to
 be sent back to the client device
         "PAYLOAD": "The message which should be received from AWS IoT Broker after
 re-connecting to a persistent session from the specified trigger topic.",

         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Persistent_Session_Happy_Case",
         "version":"0.0.0"
      }
   }
]
```

« Session persistante - Expiration de session »

Ce cas de test permet de valider le comportement de l'appareil lorsqu'un appareil déconnecté se reconnecte à une session persistante expirée. Une fois la session expirée, nous nous attendons à ce que l'appareil se réabonne aux rubriques auxquelles il était précédemment abonné en envoyant explicitement un nouveau paquet SUBSCRIBE.

Lors de la première connexion, nous nous attendons à ce que l'appareil de test SE CONNECTE au courtier AWS IoT, car son `CleanSession` indicateur est défini sur false pour lancer une session persistante. L'appareil doit ensuite s'abonner à une rubrique déclencheur. L'appareil est ensuite déconnecté par AWS IoT Core Device Advisor, après un abonnement réussi et le lancement d'une session persistante. Après la déconnexion, AWS IoT Core Device Advisor

permet au périphérique de test de se reconnecter au point de terminaison de test. À ce stade, lorsque le périphérique de test envoie un autre paquet CONNECT, AWS IoT Core Device Advisor renvoie un paquet CONNACK indiquant que la session persistante a expiré. L'appareil de test doit interpréter ce paquet correctement et il est censé se réabonner à la même rubrique déclencheur à la fin de la session persistante. Si l'appareil de test ne se réabonne pas à son déclencheur de rubrique, le scénario de test échoue. Pour que le test réussisse, l'appareil doit comprendre que la session persistante est terminée et renvoyer un nouveau paquet SUBSCRIBE pour la même rubrique de déclenchement lors de la deuxième connexion.

Si ce scénario de test réussit pour un appareil de test, cela indique que l'appareil est capable de gérer la reconnexion à l'expiration de la session persistante de la manière attendue.

Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'au moins 4 minutes. L'appareil de test doit s'abonner explicitement à un `TRIGGER_TOPIC`, auquel il n'était pas abonné auparavant. Pour réussir le scénario de test, l'appareil de test doit envoyer un paquet CONNECT avec l'indicateur `CleanSession` défini sur false et s'abonner avec succès à une rubrique de déclenchement avec une QoS 1. Une fois la connexion établie, AWS IoT Core Device Advisor déconnecte l'appareil. Après la déconnexion, AWS IoT Core Device Advisor permet à l'appareil de se reconnecter, et l'appareil devrait s'abonner à nouveau à ce service, `TRIGGER_TOPIC` car AWS IoT Core Device Advisor aurait mis fin à la session persistante.

```
"tests":[
   {
      "name":"my_expired_persistent_session_test",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Expired_Persistent_Session",
         "version":"0.0.0"
      }
```

```
   }
]
```

Shadow

Utilisez ces tests pour vérifier que vos appareils testés utilisent correctement le service AWS IoT Device Shadow. Pour plus d'informations, consultez AWS IoT Service Device Shadow. Si ces cas de test sont configurés dans votre suite de tests, il est nécessaire de fournir un élément lors du démarrage de l'exécution de la suite.

Le MQTT over n' WebSocketest pas pris en charge pour le moment.

Publication

« L'appareil publie son état après sa connexion (Happy case) »

Valide si un appareil peut publier son état après s'être connecté à AWS IoT Core

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_shadow_publish_reported_state",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME",
         "REPORTED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         }
      },
      "test":{
         "id":"Shadow_Publish_Reported_State",
         "version":"0.0.0"
```

```
      }
   }
]
```

Les REPORTED_STATE peuvent être fournis pour une validation supplémentaire de l'état shadow exact de votre appareil, une fois celui-ci connecté. Par défaut, ce scénario de test valide l'état de publication de votre appareil.

Si *SHADOW_NAME* n'est pas fourni, le scénario de test recherche par défaut les messages publiés dans les préfixes de rubrique du type Unnamed (classic) shadow. Indiquez un nom shadow si votre appareil utilise le type shadow nommé. Consultez la section Utilisation des shadows dans les appareils pour plus d'informations.

Mettre à jour

« L'appareil met à jour l'état signalé à l'état souhaité (Happy case) »

Valide si votre appareil lit tous les messages de mise à jour reçus et synchronise l'état de l'appareil pour qu'il corresponde aux propriétés d'état souhaitées. Votre appareil devrait publier son dernier état signalé après la synchronisation. Si votre appareil dispose déjà d'un shadow existant avant d'exécuter le test, assurez-vous que l'état souhaité configuré pour le scénario de test et l'état signalé existant ne correspondent pas déjà. Vous pouvez identifier les messages de mise à jour de Shadow envoyés par Device Advisor en consultant le ClientTokenchamp tel qu'il sera dans le document ShadowDeviceAdvisorShadowTestCaseSetup.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 2 minutes.

```
"tests":[
   {
      "name":"my_shadow_update_reported_state",
      "configuration": {
         "DESIRED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
```

```
         },
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME"
      },
      "test":{
         "id":"Shadow_Update_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

Le `DESIRED_STATE` doit avoir au moins un attribut et une valeur associée.

Si `SHADOW_NAME` n'est pas fourni, alors le scénario de test recherche par défaut les messages publiés dans les préfixes de rubrique du type Unnamed (classic) shadow. Indiquez un nom shadow si votre appareil utilise le type shadow nommé. Consultez la section Utilisation des shadows dans les appareils pour plus d'informations.

Exécution d’une tâche

« L'appareil peut terminer l'exécution d'une tâche »

Ce cas de test vous permet de vérifier si votre appareil est en mesure de recevoir des mises à jour à l'aide de AWS IoT Jobs et de publier l'état des mises à jour réussies. Pour plus d'informations sur les AWS IoT offres d'emploi, consultez la section Offres d'emploi.

Pour exécuter ce scénario de test avec succès, vous devez attribuer votre rôle d'appareil à deux AWS rubriques réservées. Pour vous abonner aux messages liés à l'activité professionnelle, utilisez les rubriques notify et notify-next. Le rôle de votre appareil doit autoriser l'action PUBLISH pour les rubriques suivantes :

- $aws/things/thingName/jobs/jobId/get
- $aws/things/thingName/jobs/jobId/update

Il est recommandé d'accorder les actions SUBSCRIBE et RECEIVE pour les rubriques suivantes :

- $aws/things/ ThingName/jobs/get/accepted
- $aws/things/thingName/jobs/jobId/get/rejected
- $aws/things/thingName/jobs/jobId/update/accepted

- $aws/things/thingName/jobs/jobId/update/rejected

Il est recommandé d'accorder l’action SUBSCRIBE pour la rubrique suivante :

- $aws/things/thingName/jobs/notify-next

Pour plus d'informations sur ces sujets réservés, consultez la section rubriques réservées aux AWS IoT Jobs.

Le MQTT over n' WebSocketest pas pris en charge pour le moment.

Définition du cas de test de l'API :

> **Note**
>
> `EXECUTION_TIMEOUT` a une valeur par défaut de 5 minutes. Nous recommandons une valeur de délai d'attente de 3 minutes. En fonction du document ou de la source du AWS IoT Job fourni, ajustez la valeur du délai d'attente (par exemple, si l'exécution d'une tâche prend du temps, définissez une valeur de délai d'expiration plus longue pour le scénario de test). Pour exécuter le test, un document de AWS IoT Job valide ou un ID de job déjà existant est requis. Un document AWS IoT Job peut être fourni sous forme de document JSON ou de lien S3. Si un document job est fourni, la fourniture d’un identifiant job est facultative. Si un identifiant de travail est fourni, Device Advisor l'utilisera pour créer le AWS IoT Job en votre nom. Si le document job n'est pas fourni, vous pouvez fournir un identifiant existant qui se trouve dans la même région que celle dans laquelle vous exécutez le scénario de test. Dans ce cas, Device Advisor utilisera ce AWS IoT Job lors de l'exécution du scénario de test.

```
"tests": [
   {
      "name":"my_job_execution",
      "configuration": {
         // optional:
         // Test case will create a job task by using either JOB_DOCUMENT or
 JOB_DOCUMENT_SOURCE.
         // If you manage the job task on your own, leave it empty and provide the
 JOB_JOBID (self-managed job task).
         // JOB_DOCUMENT is a JSON formatted string
         "JOB_DOCUMENT": "{
            \"operation\":\"reboot\",
```

```
            \"files\" : {
               \"fileName\" : \"install.py\",
               \"url\" : \"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/
bucket-name/key}\"
            }
         }",
         // JOB_DOCUMENT_SOURCE is an S3 link to the job document. It will be used
 only if JOB_DOCUMENT is not provided.
         "JOB_DOCUMENT_SOURCE": "https://s3.amazonaws.com/bucket-name/key",
         // JOB_JOBID is mandatory, only if neither document nor document source is
 provided. (Test case needs to know the self-managed job task id).
         "JOB_JOBID": "String",
         // JOB_PRESIGN_ROLE_ARN is used for the presign Url, which will replace the
 placeholder in the JOB_DOCUMENT field
         "JOB_PRESIGN_ROLE_ARN": "String",
         // Presigned Url expiration time. It must be between 60 and 3600 seconds,
 with the default value being 3600.
         "JOB_PRESIGN_EXPIRES_IN_SEC": "Long"
         "EXECUTION_TIMEOUT": "300", // in seconds
      },
      "test": {
         "id": "Job_Execution",
         "version": "0.0.0"
      }
   }
]
```

Pour plus d'informations sur la création et l'utilisation de documents job, consultez document job.

Autorisations et politiques

Vous pouvez utiliser les tests suivants pour déterminer si les politiques associées aux certificats de vos appareils respectent les meilleures pratiques standard.

Le MQTT over n' WebSocketest pas pris en charge pour le moment.

« Les politiques associées aux certificats de l'appareil ne contiennent pas de caractères génériques »

Valide si les politiques d'autorisation associées à un appareil respectent les meilleures pratiques et n'accordent pas à l'appareil plus d'autorisations que nécessaire.

Définition du cas de test de l'API :

> **Note**
>
> EXECUTION_TIMEOUT a une valeur par défaut de 1 minute. Nous vous recommandons de définir un délai d'au moins 30 secondes.

```
"tests":[
   {
        "name":"my_security_device_policies",
        "configuration": {
            // optional:
            "EXECUTION_TIMEOUT":"60"    // in seconds
        },
        "test": {
            "id": "Security_Device_Policies",
            "version": "0.0.0"
        }
    }
]
```

Tests de longue durée

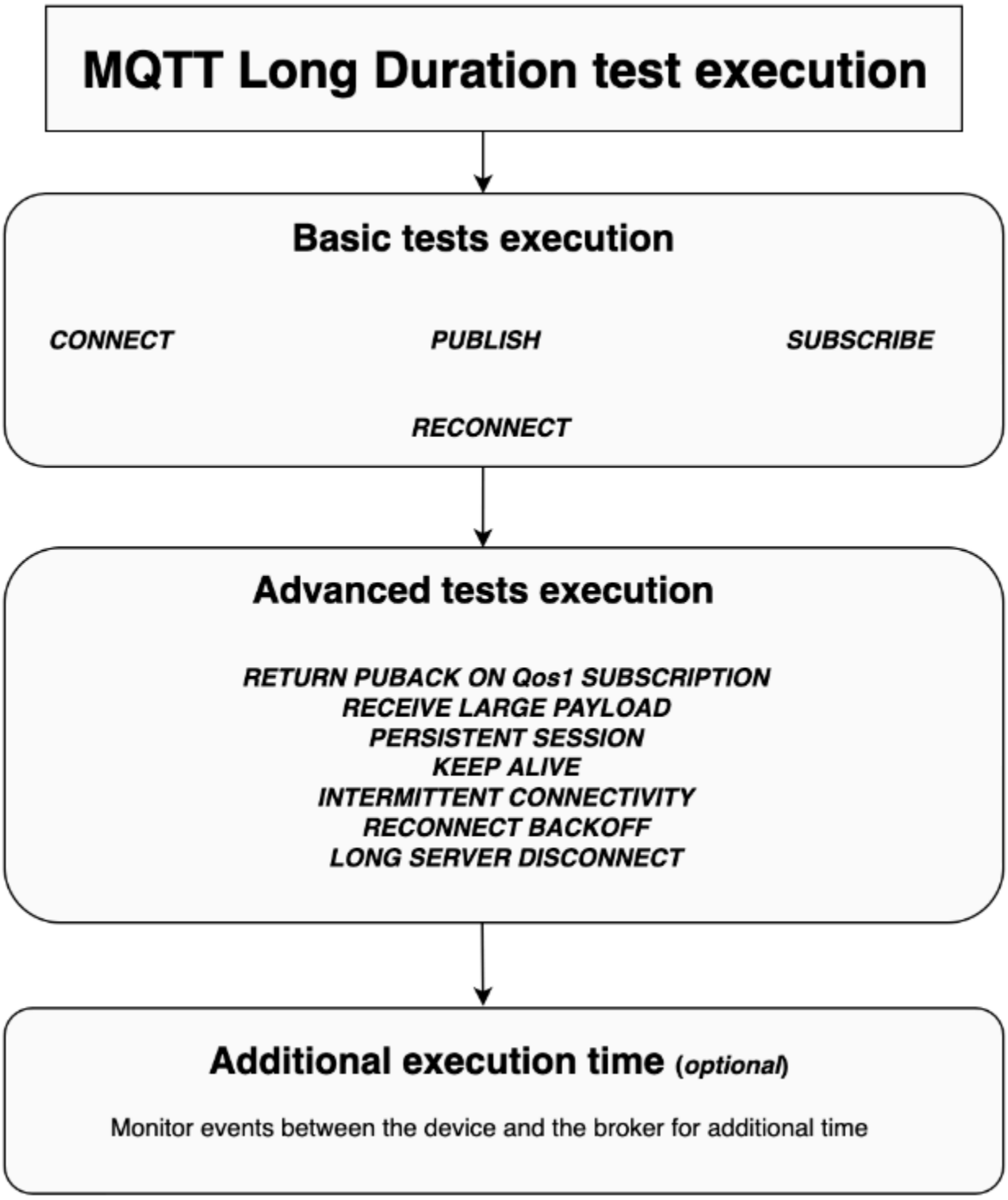
Les tests de longue durée sont une nouvelle suite de tests qui surveille le comportement d'un appareil lorsqu'il fonctionne sur de longues périodes. Comparé à l'exécution de tests individuels axés sur des comportements spécifiques d'un appareil, le test de longue durée examine le comportement de l'appareil dans divers scénarios réels au cours de sa durée de vie. Device Advisor orchestre les tests dans l'ordre le plus efficace possible. Le test génère des résultats et des journaux, y compris un journal récapitulatif contenant des mesures utiles sur les performances de l'appareil pendant le test.

Cas de test MQTT de longue durée

Dans le cas de test MQTT de longue durée, le comportement de l'appareil est initialement observé dans des scénarios de cas heureux tels que MQTT Connect, Subscribe, Publish et Reconnect. Ensuite, l'appareil est observé dans plusieurs scénarios de défaillance complexes tels que l'interruption de reconnexion MQTT, la déconnexion longue du serveur et la connectivité intermittente.

Flux d'exécution des scénarios de test MQTT de longue durée

L'exécution d'un scénario de test MQTT de longue durée comporte trois phases :


MQTT Long Duration test execution
Basic tests execution
CONNECT
PUBLISH
SUBSCRIBE
RECONNECT
Advanced tests execution
RETURN PUBACK ON Qos1 SUBSCRIPTION
RECEIVE LARGE PAYLOAD
PERSISTENT SESSION
KEEP ALIVE
INTERMITTENT CONNECTIVITY
RECONNECT BACKOFF
LONG SERVER DISCONNECT
Additional execution time (optional)
Monitor events between the device and the broker for additional time

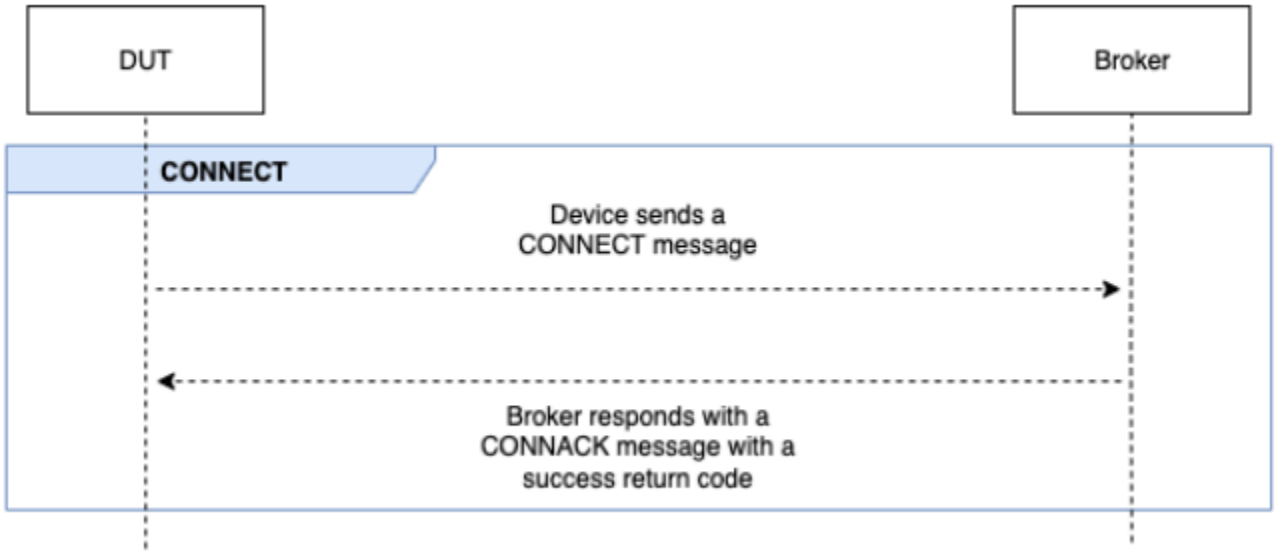
Exécution de tests de base

Dans cette phase, le scénario de test exécute des tests simples en parallèle. Le test valide si l'appareil dispose des opérations sélectionnées dans la configuration.

L'ensemble de tests de base peut inclure les éléments suivants, en fonction des opérations sélectionnées :

CONNECT

Ce scénario permet de vérifier si l'appareil est capable d'établir une connexion réussie avec le courtier.

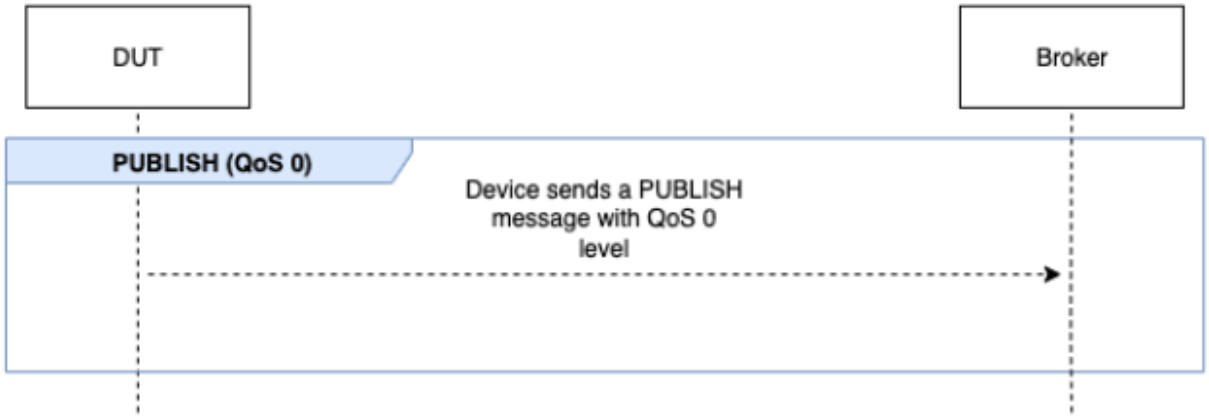


PUBLISH

Ce scénario permet de vérifier si l'appareil publie avec succès auprès du courtier.

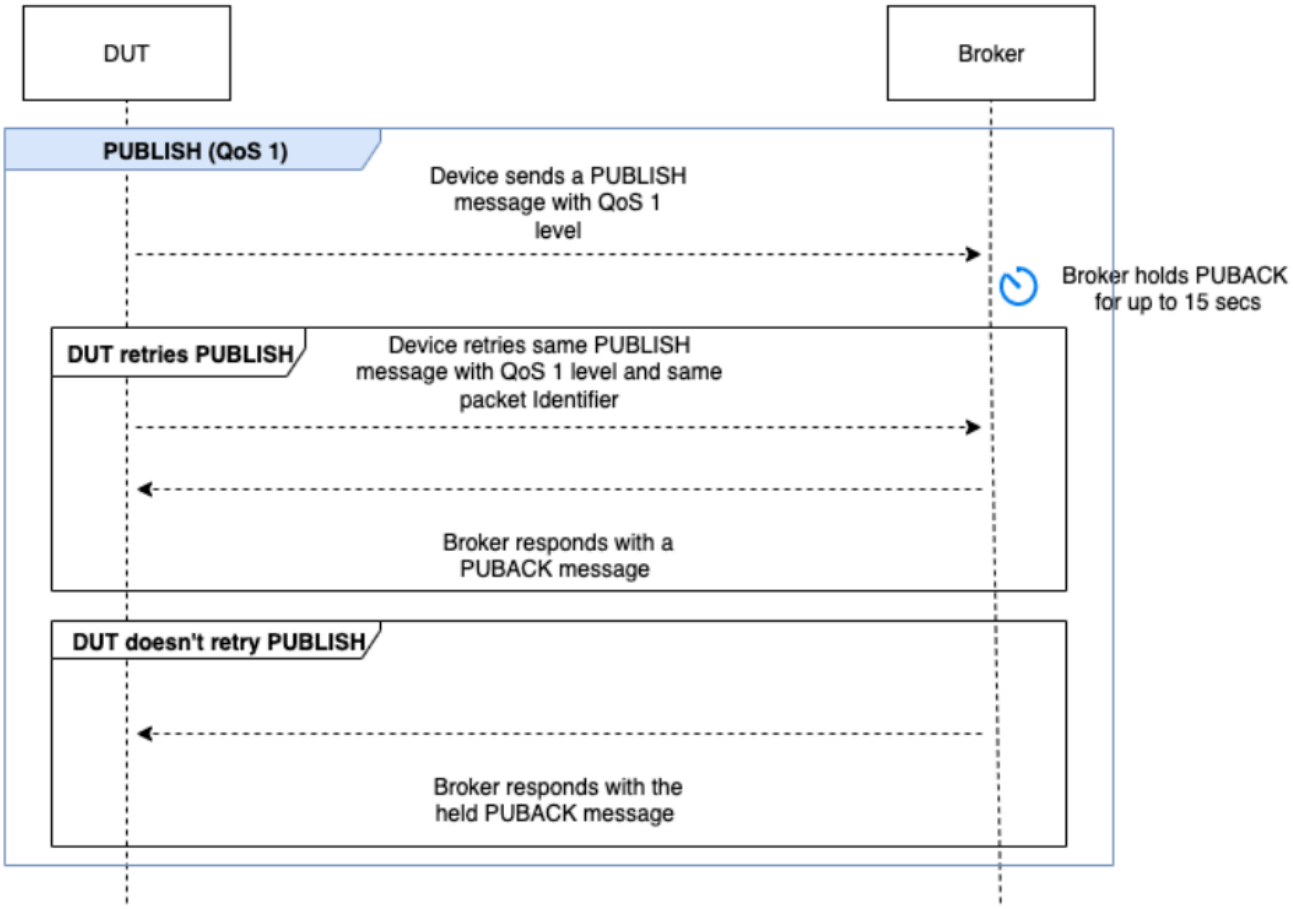
QoS 0

Ce cas de test valide si l'appareil envoie avec succès un message `PUBLISH` au courtier lors d'une publication avec QoS 0. Le test n'attend pas que le message `PUBACK` soit reçu par l'appareil.



QoS 1

Dans ce cas de test, l'appareil devrait envoyer deux messages `PUBLISH` au courtier avec QoS 1. Après le premier message `PUBLISH`, le courtier attend jusqu'à 15 secondes avant de répondre. L'appareil doit réessayer le message `PUBLISH` d'origine avec le même identifiant de paquet dans le délai de 15 secondes. Si c'est le cas, le courtier répond par un message `PUBACK` et le test est validé. Si l'appareil ne réessaie pas `PUBLISH`, `PUBACK`initial lui est envoyé et le test est marqué comme réussi avec des avertissements, ainsi qu'un message système. Pendant l'exécution du test, si l'appareil perd la connexion et se reconnecte, le scénario de test sera réinitialisé sans échec et l'appareil devra effectuer à nouveau les étapes du scénario de test.

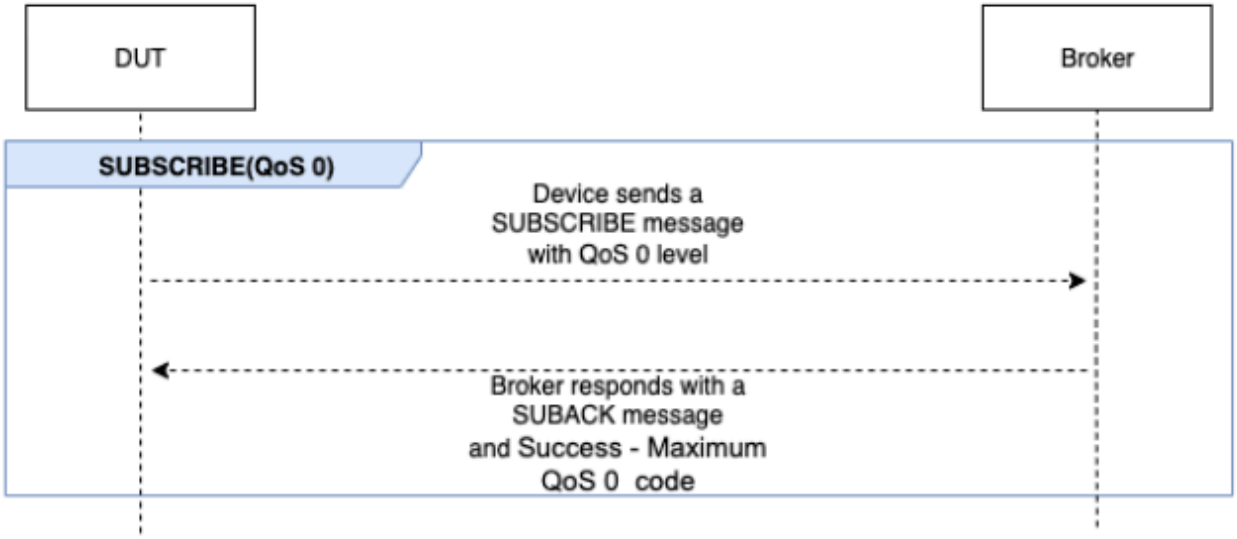


SUBSCRIBE

Ce scénario valide si l'appareil s'abonne avec succès auprès du courtier.

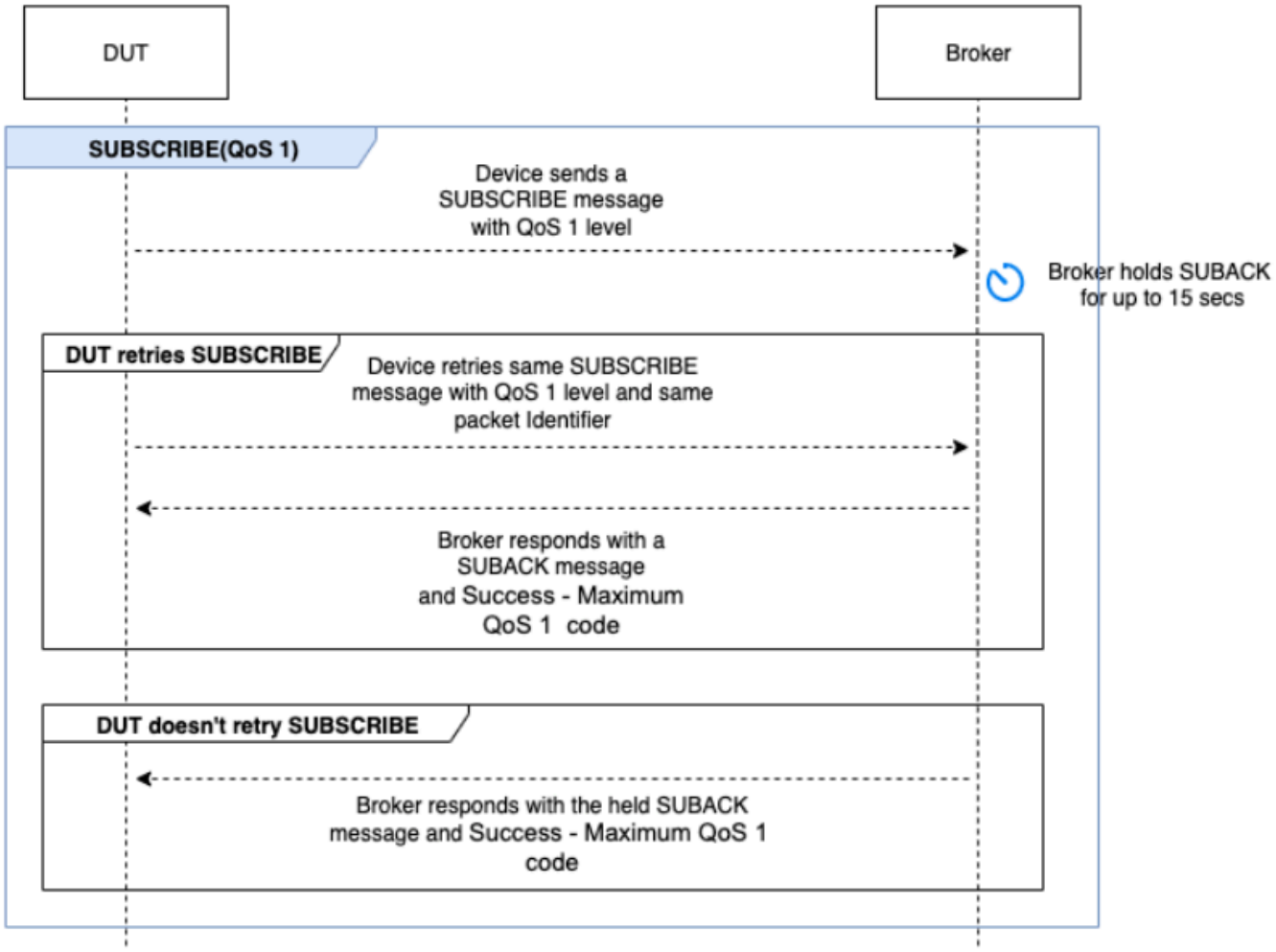
QoS 0

Ce cas de test valide si l'appareil envoie avec succès un message `SUBSCRIBE` au courtier lors d'un abonnement avec QoS 0. Le test n'attend pas que l'appareil reçoive un message SUBACK.



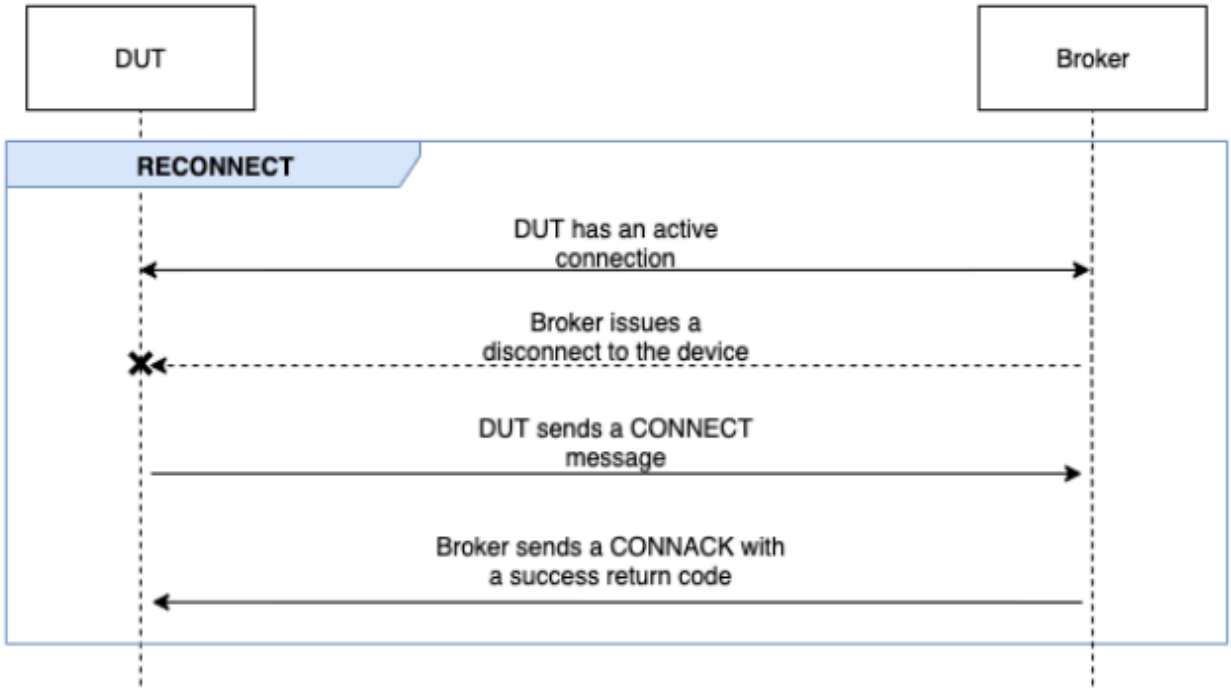
QoS 1

Dans ce cas de test, l'appareil devrait envoyer deux messages `SUBSCRIBE` au courtier avec QoS 1. Après le premier message `SUBSCRIBE`, le courtier attend jusqu'à 15 secondes avant de répondre. L'appareil doit réessayer le message `SUBSCRIBE` d'origine avec le même identifiant de paquet dans le délai de 15 secondes. Si c'est le cas, le courtier répond par un message `SUBACK` et le test est validé. Si l'appareil ne réessaie pas `SUBSCRIBE`, `SUBACK`initial lui est envoyé et le test est marqué comme réussi avec des avertissements, ainsi qu'un message système. Pendant l'exécution du test, si l'appareil perd la connexion et se reconnecte, le scénario de test sera réinitialisé sans échec et l'appareil devra effectuer à nouveau les étapes du scénario de test.



RECONNECT

Ce scénario vérifie si l'appareil se reconnecte avec succès au courtier une fois que l'appareil est déconnecté d'une connexion réussie. Device Advisor ne déconnecte pas l'appareil s'il s'est connecté plusieurs fois au cours de la suite de tests. Au lieu de cela, il marquera le test comme réussi.



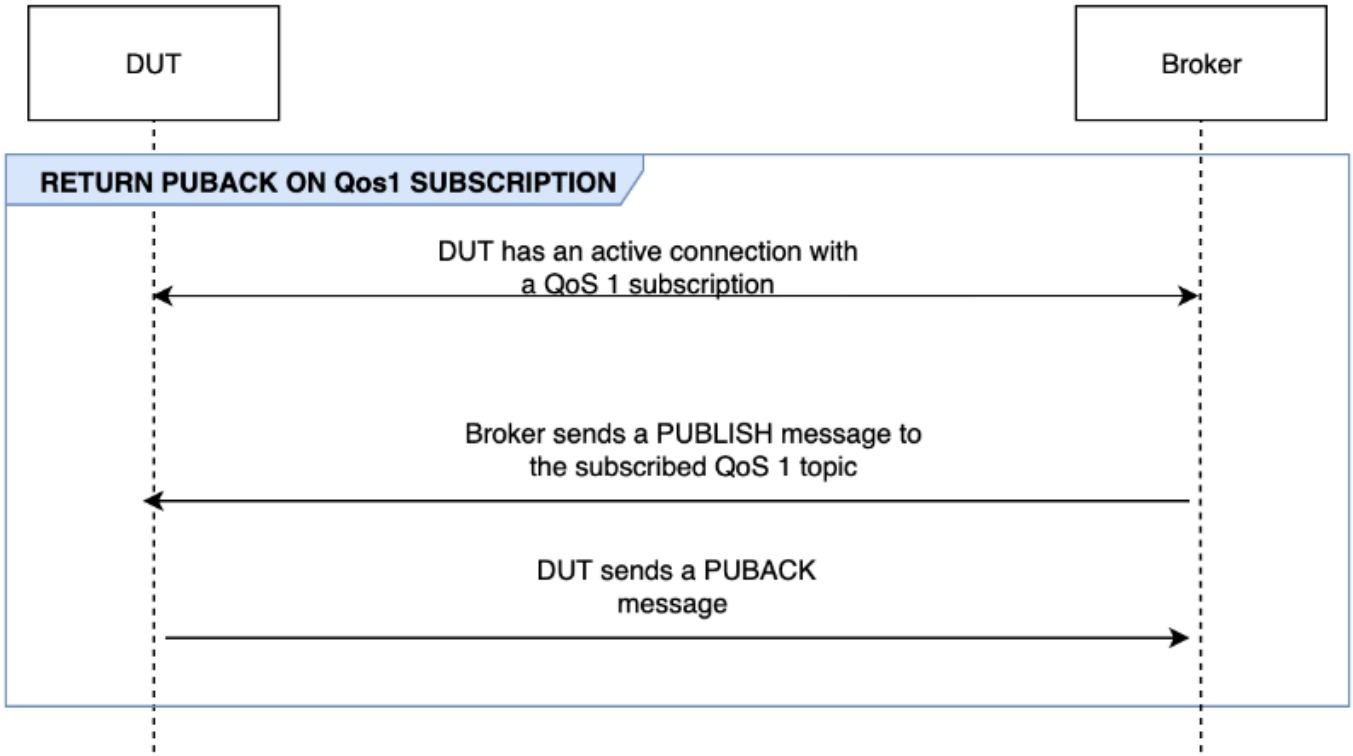
Exécution de tests avancés

Au cours de cette phase, le scénario de test exécute des tests plus complexes en série pour valider si le dispositif suit les meilleures pratiques. Ces tests avancés peuvent être sélectionnés et peuvent être désactivés s'ils ne sont pas nécessaires. Chaque test avancé possède sa propre valeur de délai d'attente en fonction des exigences du scénario.

RETURN PUBACK ON QoS 1 SUBSCRIPTION

> **Note**
>
> Sélectionnez ce scénario uniquement si votre appareil est capable d'exécuter des abonnements QoS 1.

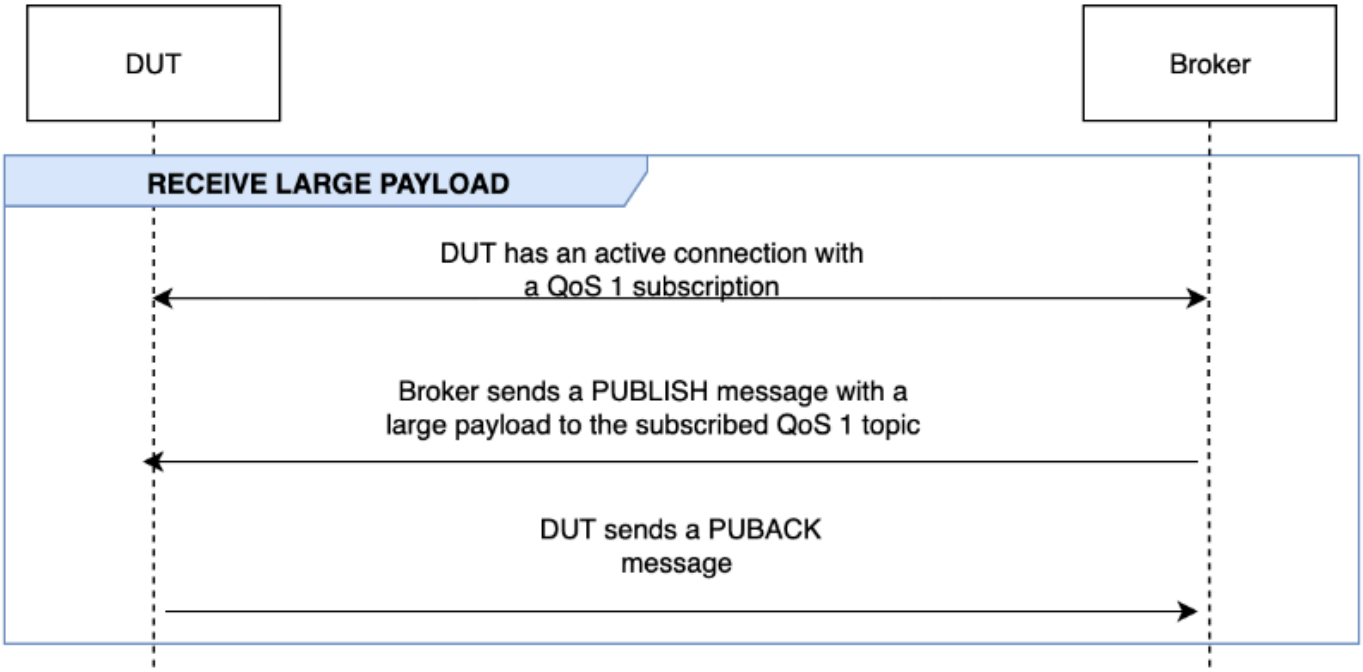
Ce scénario valide si, une fois que l'appareil s'est abonné à une rubrique et a reçu un `PUBLISH` message du courtier, il renvoie un `PUBACK` message.



RECEIVE LARGE PAYLOAD

> **Note**
>
> Sélectionnez ce scénario si votre appareil est capable d'exécuter des abonnements QoS 1.

Ce scénario valide si l'appareil répond par un `PUBACK` message après avoir reçu un `PUBLISH` message du courtier pour un sujet QoS 1 avec une charge utile importante. Le format de la charge utile attendue peut être configuré à l'aide de l'option `LONG_PAYLOAD_FORMAT`.



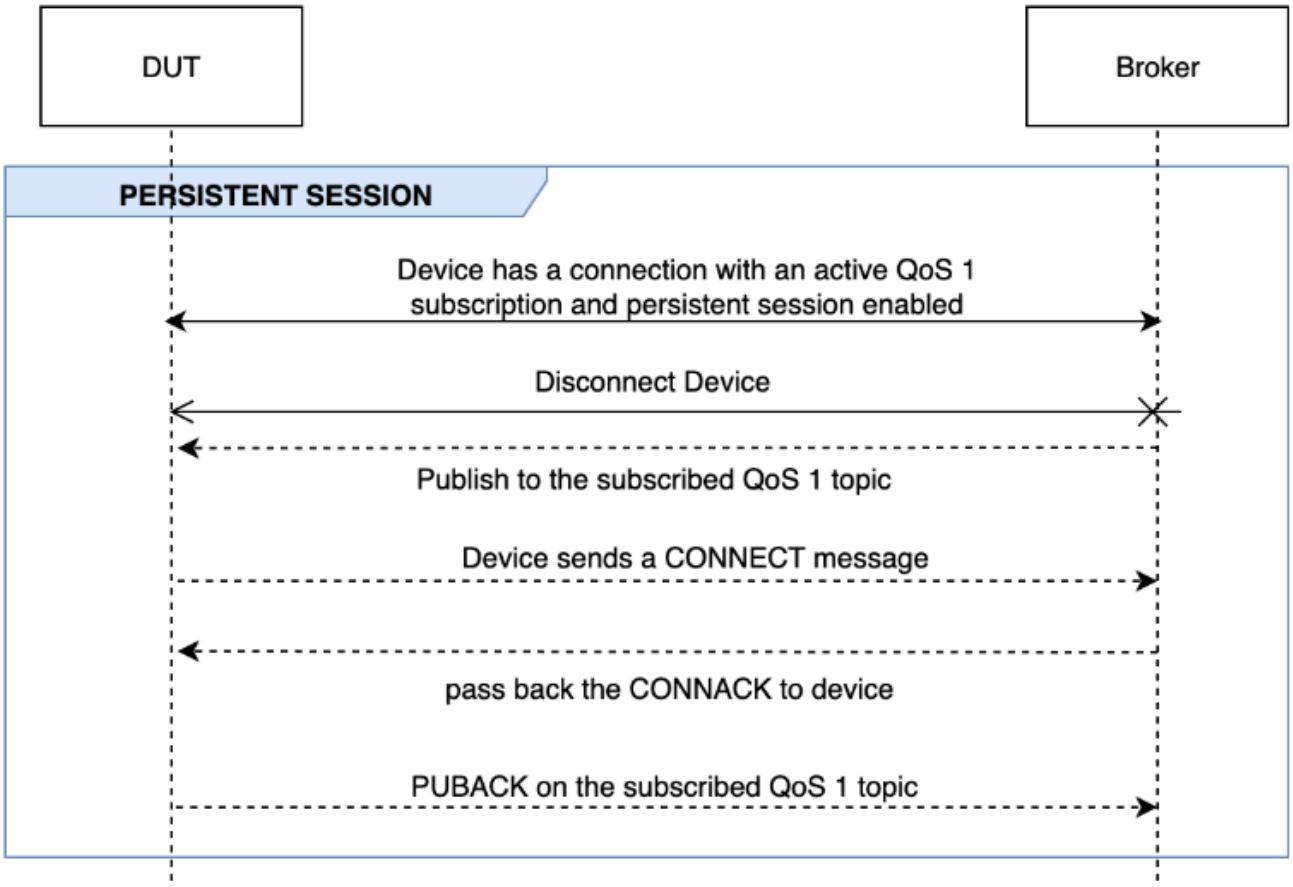
PERSISTENT SESSION

> **Note**
>
> Sélectionnez ce scénario uniquement si votre appareil est capable d'effectuer des abonnements QoS 1 et peut maintenir une session persistante.

Ce scénario valide le comportement de l'appareil lors du maintien de sessions persistantes. Le test valide lorsque les conditions suivantes sont réunies :

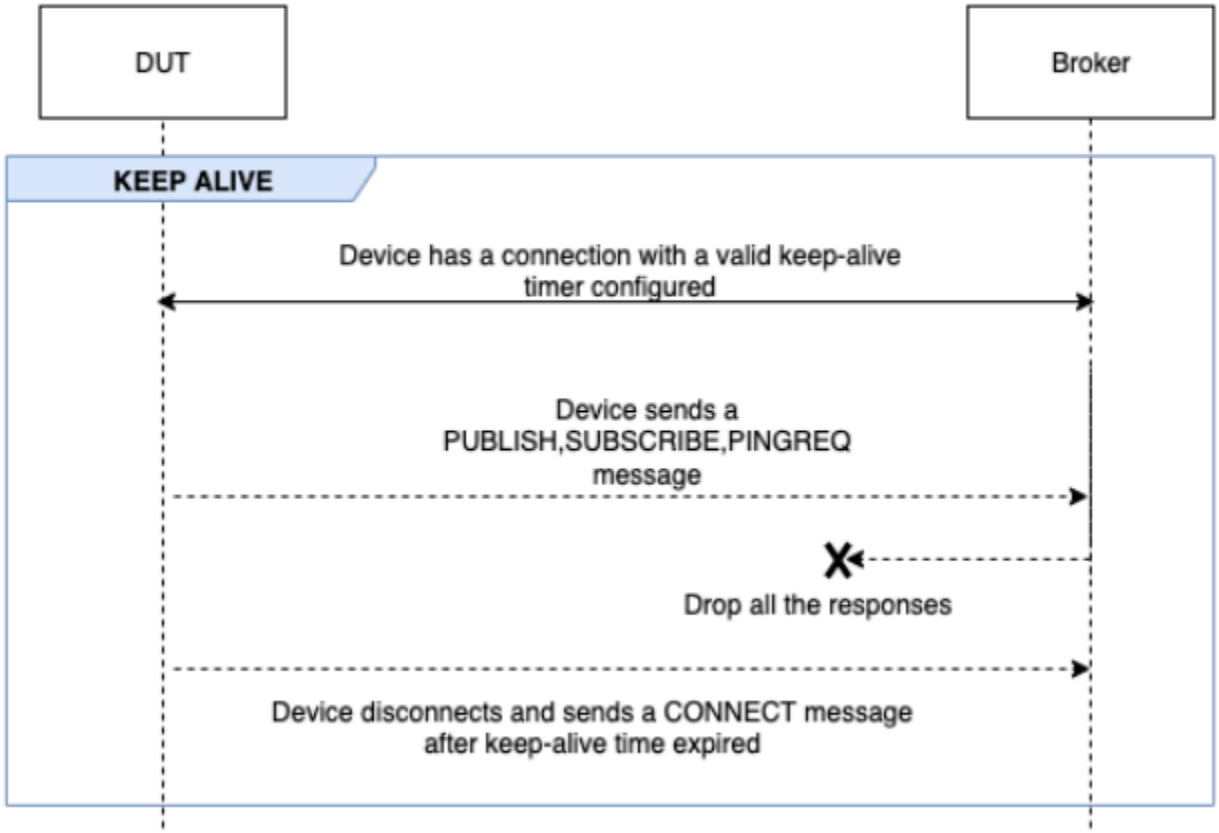
- L'appareil se connecte au courtier avec un abonnement QoS 1 actif et des sessions persistantes activées.
- L'appareil se déconnecte correctement du courtier pendant la session.
- L'appareil se reconnecte au courtier et reprend les abonnements à ses rubriques de déclenchement sans se réabonner explicitement à ces rubriques.
- L'appareil reçoit avec succès les messages stockés par le courtier pour les sujets auxquels il est abonné et fonctionne comme prévu.

Pour plus d'informations sur les sessions AWS IoT persistantes, consultez la section Utilisation des sessions persistantes MQTT.



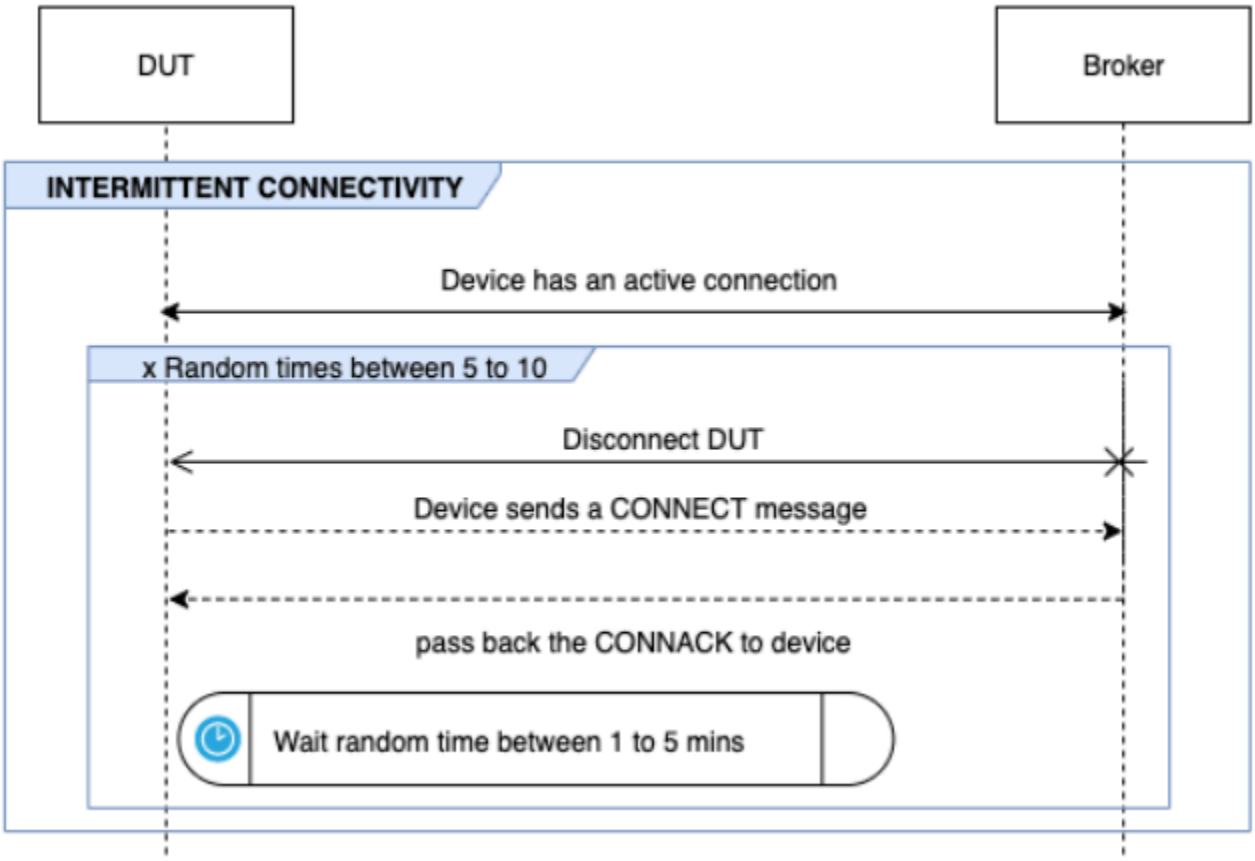
KEEP ALIVE

Ce scénario vérifie si l'appareil se déconnecte correctement après avoir reçu une réponse ping du courtier. La connexion doit avoir une minuterie de maintien valide configurée. Dans le cadre de ce test, le courtier bloque toutes les réponses envoyées pour `PUBLISHSUBSCRIBE`, et les `PINGREQ` messages. Il vérifie également si l'appareil testé déconnecte la connexion MQTT.



INTERMITTENT CONNECTIVITY

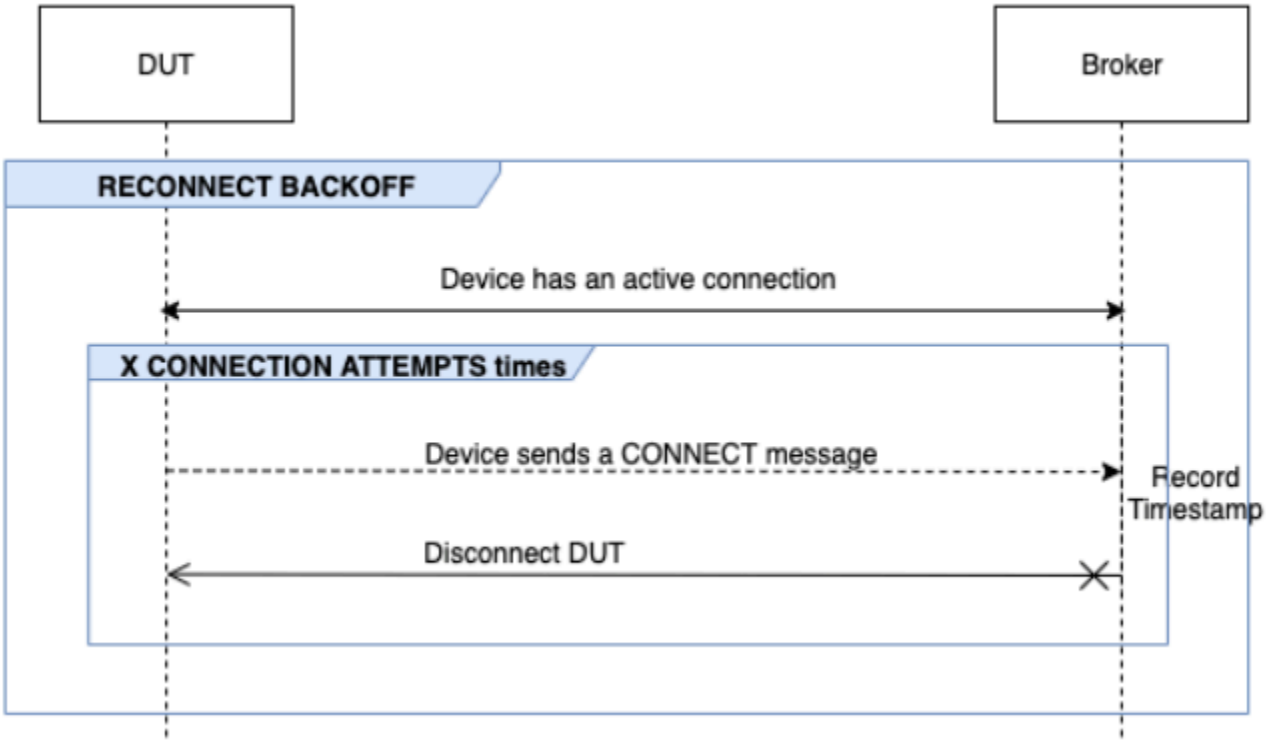
Ce scénario valide si l'appareil peut se reconnecter au courtier après que celui-ci l'ait déconnecté à intervalles aléatoires pendant une période de temps aléatoire.



RECONNECT BACKOFF

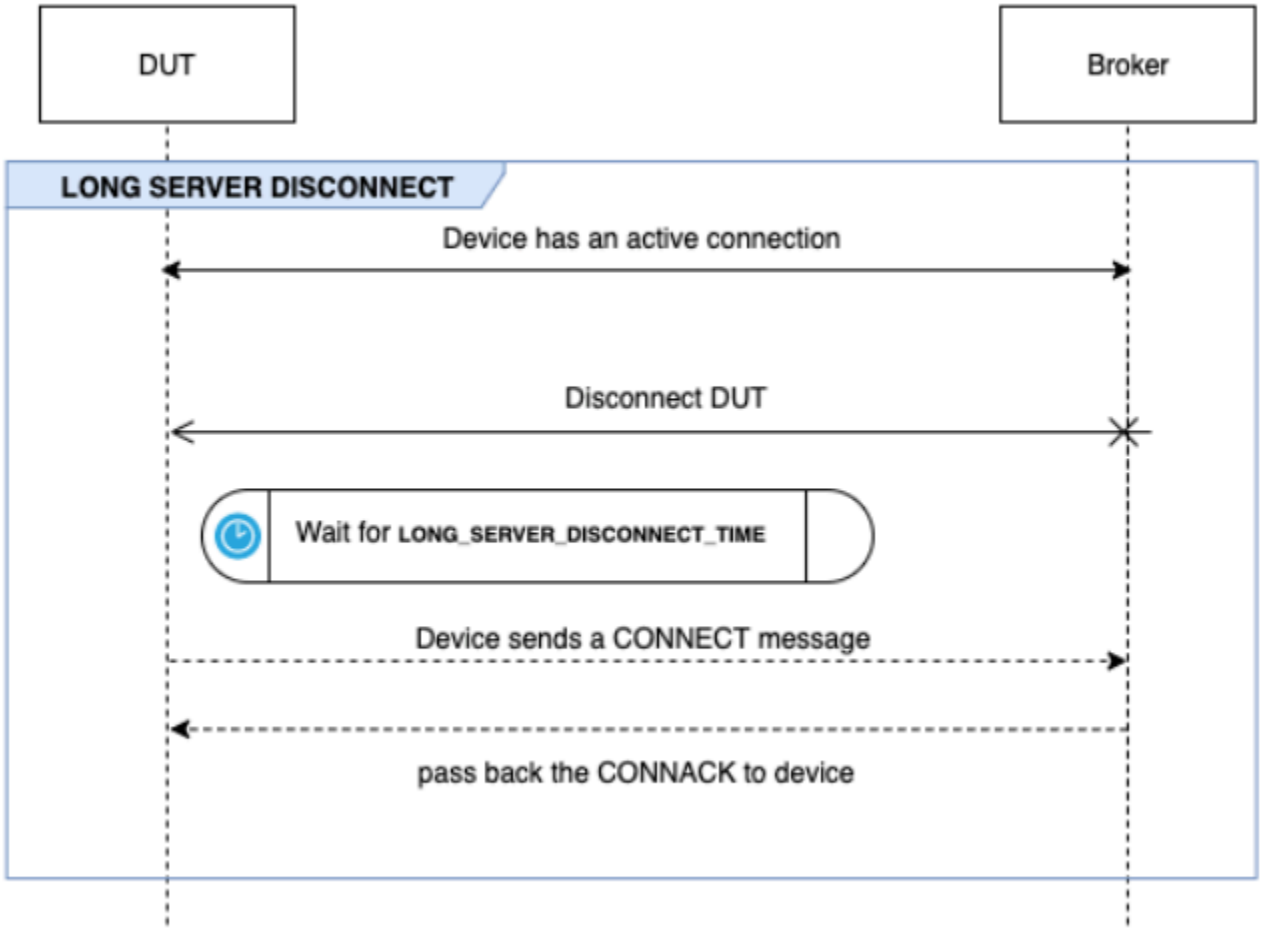
Ce scénario valide si l'appareil dispose d'un mécanisme de sauvegarde mis en œuvre lorsque le courtier s'en déconnecte plusieurs fois. Device Advisor signale le type d'intervalle comme exponentiel, instabilité, linéaire ou constant. Le nombre de tentatives d'interruption est configurable à l'aide de l'option `BACKOFF_CONNECTION_ATTEMPTS`. La valeur par défaut est 5. La valeur est configurable entre 5 et 10.

Pour réussir ce test, nous vous recommandons d'implémenter Backoff exponentiel et Gigue sur l'appareil testé dans ce test.



LONG SERVER DISCONNECT

Ce scénario vérifie si l'appareil peut se reconnecter avec succès après que le courtier l'a déconnecté pendant une longue période (jusqu'à 120 minutes). L'heure de déconnexion du serveur peut être configurée à l'aide de l'option `LONG_SERVER_DISCONNECT_TIME`. La valeur par défaut est de 120 minutes. Cette valeur est configurable entre 30 et 120 minutes.



Temps d'exécution supplémentaire

Le temps d'exécution supplémentaire est le temps pendant lequel le test attend après avoir terminé tous les tests ci-dessus et avant de terminer le scénario de test. Les clients utilisent cette période supplémentaire pour surveiller et enregistrer toutes les communications entre l'appareil et le courtier. Le temps d'exécution supplémentaire peut être configuré à l'aide de l'option `ADDITIONAL_EXECUTION_TIME`. Par défaut, cette option est définie sur 0 minute et peut aller de 0 à 120 minutes.

Options de configuration des tests de longue durée MQTT

Toutes les options de configuration fournies pour le test de longue durée MQTT sont facultatives. Les options suivantes sont disponibles :

OPERATIONS

La liste des opérations effectuées par le périphérique, telles que`CONNECT`, `PUBLISH` et `SUBSCRIBE`. Le scénario de test exécute des scénarios basés sur les opérations spécifiées. Les opérations qui ne sont pas spécifiées sont considérées comme valides.

```
{
"OPERATIONS": ["PUBLISH", "SUBSCRIBE"]
//by default the test assumes device can CONNECT
}
```

SCENARIOS

Sur la base des opérations sélectionnées, le scénario de test exécute des scénarios pour valider le comportement de l'appareil. Il existe deux types de scénarios :

- Les scénarios de base sont des tests simples qui valident si le périphérique peut effectuer les opérations sélectionnées ci-dessus dans le cadre de la configuration. Ils sont présélectionnés en fonction des opérations spécifiées dans la configuration. Aucune autre saisie n'est requise dans la configuration.
- Les scénarios avancés sont des scénarios plus complexes qui sont exécutés par rapport à l'appareil pour valider si celui-ci suit les meilleures pratiques lorsqu'il est confronté à des conditions réelles. Ils sont facultatifs et peuvent être transmis sous forme de tableau de scénarios à l'entrée de configuration de la suite de tests.

```
{
    "SCENARIOS": [      // list of advanced scenarios
                "PUBACK_QOS_1",
                "RECEIVE_LARGE_PAYLOAD",
                "PERSISTENT_SESSION",
                "KEEP_ALIVE",
                "INTERMITTENT_CONNECTIVITY",
                "RECONNECT_BACK_OFF",
                "LONG_SERVER_DISCONNECT"
    ]
}
```

BASIC_TESTS_EXECUTION_TIME_OUT:

Durée maximale pendant laquelle le scénario de test attendra la fin de tous les tests de base. La valeur par défaut est de 60 minutes. Cette valeur est configurable entre 30 et 120 minutes.

LONG_SERVER_DISCONNECT_TIME:

Temps nécessaire au scénario de test pour déconnecter et reconnecter l'appareil pendant le test de déconnexion longue du serveur. La valeur par défaut est de 60 minutes. Cette valeur est configurable entre 30 et 120 minutes.

ADDITIONAL_EXECUTION_TIME:

La configuration de cette option fournit une fenêtre temporelle une fois tous les tests terminés, afin de surveiller les événements entre l'appareil et le courtier. La valeur par défaut est de 0 minutes. Cette valeur est configurable entre 0 et 120 minutes.

BACKOFF_CONNECTION_ATTEMPTS:

Cette option configure le nombre de fois que l'appareil est déconnecté par le scénario de test. Ceci est utilisé par le test Reconnect Backoff. La valeur par défaut est de 5 tentatives. Cette valeur est configurable entre 5 et 10.

LONG_PAYLOAD_FORMAT:

Format de la charge utile du message attendu par l'appareil lorsque le scénario de test est publié dans une rubrique QoS 1 à laquelle l'appareil est abonné.

Définition du cas de test de l'API :

```
{
"tests":[
   {
      "name":"my_mqtt_long_duration_test",
      "configuration": {
         // optional
         "OPERATIONS": ["PUBLISH", "SUBSCRIBE"],
         "SCENARIOS": [
            "LONG_SERVER_DISCONNECT",
            "RECONNECT_BACK_OFF",
            "KEEP_ALIVE",
            "RECEIVE_LARGE_PAYLOAD",
            "INTERMITTENT_CONNECTIVITY",
            "PERSISTENT_SESSION",
         ],
         "BASIC_TESTS_EXECUTION_TIMEOUT": 60, // in minutes (60 minutes by default)
         "LONG_SERVER_DISCONNECT_TIME": 60,   // in minutes (120 minutes by default)
         "ADDITIONAL_EXECUTION_TIME": 60,     // in minutes (0 minutes by default)
```

```
         "BACKOFF_CONNECTION_ATTEMPTS": "5",
         "LONG_PAYLOAD_FORMAT":"{"message":"${payload}"}"
      },
      "test":{
         "id":"MQTT_Long_Duration",
         "version":"0.0.0"
      }
   }
 ]
}
```

Journal récapitulatif du scénario de test MQTT de longue durée

Le scénario de test de longue durée MQTT s'exécute sur une plus longue durée que les scénarios de test classiques. Un journal récapitulatif distinct est fourni, qui répertorie les événements importants tels que les connexions des appareils, la publication et l'abonnement pendant l'exécution. Les détails incluent ce qui a été testé, ce qui n'a pas été testé et ce qui a échoué. À la fin du journal, le test inclut un résumé de tous les événements survenus pendant l'exécution du scénario de test. Cela consiste notamment à :

- Le minuteur Keep Alive est configuré sur l'appareil.
- Indicateur de session persistante configuré sur l'appareil.
- Le nombre de connexions de l'appareil pendant le test.
- Type d'interruption de reconnexion de l'appareil, s'il est validé pour le test d'interruption de reconnexion.
- Rubriques sur lesquelles l'appareil a publié, lors de l'exécution du scénario de test.
- Rubriques sur lesquelles l'appareil s'est abonné pendant l'exécution du scénario de test.

AWS IoT Core Emplacement de l'appareil

Avant d'utiliser la fonction de localisation de l' AWS IoT Core appareil, consultez les conditions générales de cette fonctionnalité. Notez que les paramètres de votre demande de recherche de géolocalisation, tels que les données de localisation utilisées pour effectuer les recherches, et d'autres informations AWS peuvent être transmises au fournisseur de données tiers que vous avez choisi, qui peuvent être autres Région AWS que celui que vous utilisez actuellement. Le fournisseur tiers et le solveur à utiliser sont choisis en fonction de la charge utile d'entrée reçue. Pour plus d'informations, consultez Conditions de service AWS.

Utilisez la localisation des AWS IoT Core appareils pour tester l'emplacement de vos appareils IoT à l'aide de solveurs tiers. Les solveurs sont des algorithmes fournis par des fournisseurs tiers qui résolvent les données de mesure et estiment l'emplacement de votre appareil. En identifiant l'emplacement de vos appareils, vous pouvez les suivre et les déboguer sur le terrain pour résoudre les problèmes éventuels.

Les données de mesure collectées à partir de diverses sources sont résolues et les informations de géolocalisation sont rapportées sous forme de charge utile GeoJSON. Le format GeoJSON est un format utilisé pour coder les structures de données géographiques. La charge utile contient les coordonnées de latitude et de longitude de l'emplacement de votre appareil, qui sont basées sur le système de coordonnées du système géodésique mondial (). WGS84

Rubriques

- Types de mesures et solveurs
- Comment fonctionne la localisation des AWS IoT Core appareils
- Comment utiliser la localisation de AWS IoT Core l'appareil
- Résolution de la localisation des appareils IoT
- Résolution de la localisation de l'appareil à l'aide des rubriques MQTT AWS IoT Core relatives à la localisation
- Solveurs de localisation et charge utile de l'appareil

Types de mesures et solveurs

AWS IoT Core Device Location s'associe à des fournisseurs tiers pour résoudre les données de mesure et fournir une estimation de l'emplacement de l'appareil. Le tableau suivant indique les types de mesures et les résolveurs de localisation tiers, ainsi que des informations sur les appareils pris en charge. Pour plus d'informations sur les périphériques LoRa WAN et la configuration de leur emplacement, voir Configuration de la position des ressources LoRa WAN.

> **Note**
>
> Les appareils IoT généraux et les appareils Sidewalk peuvent utiliser les rubriques MQTT de localisation des appareils pour obtenir les informations de localisation. Pour les types de mesure d'adresses Wi-Fi, cellulaire et IP, si les appareils publient les données de mesure dans les rubriques réservées au format GeoJSON défini, l'emplacement du AWS IoT Core périphérique peut résoudre la position de l'appareil. Pour le type de mesure GNSS, l'appareil doit disposer de la LR11xx puce permettant de scanner les données de mesure afin d'obtenir les informations de localisation résolues à l'aide du solveur GNSS. Pour plus d'informations sur l'obtention d'informations de localisation pour les périphériques LoRa WAN, consultez la section Configuration de la position des ressources LoRa WAN dans la AWS IoT Wireless documentation.

Types de mesures et solveurs

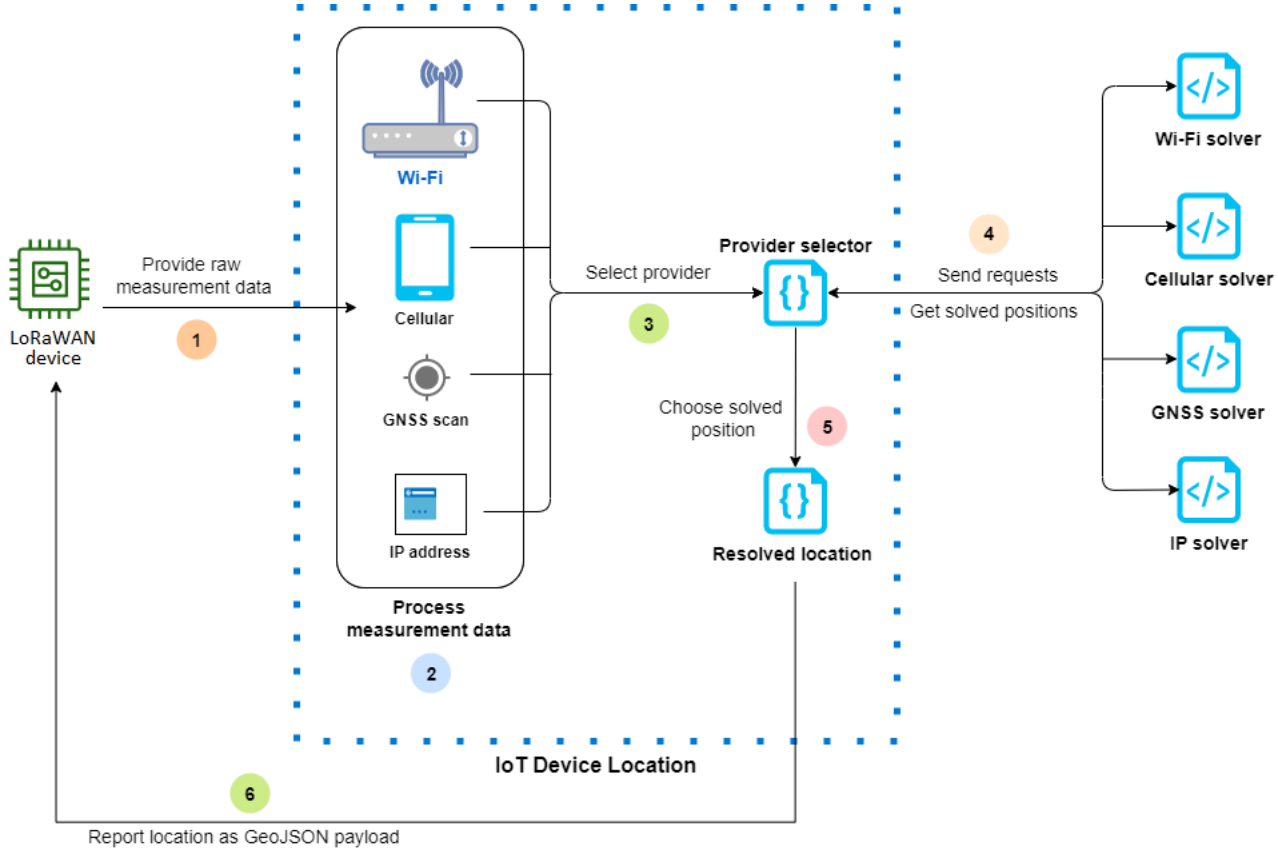
Type de mesure	Solveurs tiers	Appareils pris en charge
Points d'accès Wi-Fi	Solveur basé sur le Wi-Fi	Appareils IoT généraux, LoRa WAN et appareils Sidewalk
Tours de téléphonie cellulaire : données GSM, LTE, CDMA, SCDMA, WCMDA et TD-SCDMA	Solveur cellulaire	Appareils IoT généraux, LoRa WAN et appareils Sidewalk
Adresse IP	Solveur de recherche inversée IP	Appareils IoT généraux et appareils Sidewalk

Type de mesure	Solveurs tiers	Appareils pris en charge
Données de numérisation GNSS (messages NAV)	solveur GNSS	Appareils IoT généraux, LoRa WAN et appareils

Pour plus d'informations sur les solveurs de localisation et des exemples illustrant la charge utile de l'appareil pour les différents types de mesures, consultez Solveurs de localisation et charge utile de l'appareil.

Comment fonctionne la localisation des AWS IoT Core appareils

Le schéma suivant montre comment AWS IoT Core Device Location collecte les données de mesure et résout les informations de localisation de vos appareils.



Les étapes suivantes montrent comment fonctionne la localisation de l' AWS IoT Core appareil.

1. Recevoir des données de mesure

   Les données de mesure brutes relatives à l'emplacement de votre appareil sont d'abord envoyées depuis l'appareil. Les données de mesure sont spécifiées sous forme de charge utile JSON.

2. Données de mesure de processus

   Les données de mesure sont traitées et AWS IoT Core Device Location choisit les données de mesure à utiliser, qui peuvent être des informations Wi-Fi, cellulaires, de scan GNSS ou d'adresse IP.

3. Choisissez un solveur

   Le solveur tiers est choisi en fonction des données de mesure. Par exemple, si les données de mesure contiennent des informations sur le Wi-Fi et l'adresse IP, il choisit le solveur Wi-Fi et le solveur de recherche inversée IP.

4. Obtenir un emplacement résolu

   Une demande d'API est envoyée aux fournisseurs de solveur pour leur demander de résoudre l'emplacement. AWS IoT Core Device Location obtient ensuite les informations de géolocalisation estimées auprès des solveurs.

5. Choisissez l'emplacement résolu

   Les informations de localisation résolues et leur exactitude sont comparées, et AWS IoT Core Device Location choisit les résultats de géolocalisation avec la plus grande précision.

6. Informations sur l'emplacement de sortie

   Les informations de géolocalisation vous sont envoyées sous forme de charge utile GeoJSON. La charge utile contient les WGS84 coordonnées géographiques, les informations de précision, les niveaux de confiance et l'horodatage auquel l'emplacement résolu a été obtenu.

Comment utiliser la localisation de AWS IoT Core l'appareil

Les étapes suivantes montrent comment utiliser la localisation de AWS IoT Core l'appareil.

1. Fournir des données de mesure

   Spécifiez les données de mesure brutes relatives à l'emplacement de votre appareil sous forme de charge utile JSON. Pour récupérer les données de mesure de la charge utile, accédez aux

journaux de votre appareil ou utilisez CloudWatch Logs et copiez les informations relatives aux données de charge utile. La charge utile JSON doit contenir un ou plusieurs types de mesure de données. Pour des exemples illustrant le format de charge utile de différents solveurs, consultez Solveurs de localisation et charge utile de l'appareil.

2. Résoudre les informations de localisation

   À l'aide de la page de localisation de l'appareil de la AWS IoT console ou de l'opération GetPositionEstimateAPI, transmettez les données de mesure de la charge utile et déterminez l'emplacement de l'appareil. AWS IoT Core Device Location choisit ensuite le solveur le plus précis et indique l'emplacement de l'appareil. Pour de plus amples informations, veuillez consulter Résolution de la localisation des appareils IoT.

3. Copier les informations de localisation

   Vérifiez les informations de géolocalisation qui ont été résolues par AWS IoT Core Device Location et signalées sous forme de charge utile GeoJSON. Vous pouvez copier la charge utile pour l'utiliser avec vos applications et d'autres applications Service AWS. Par exemple, vous pouvez envoyer vos données de localisation géographique à Amazon Location Service à l'aide de l'action de Emplacement AWS IoT règle.

Les rubriques suivantes montrent comment utiliser la localisation des AWS IoT Core appareils et des exemples de charge utile de localisation des appareils.

- Résolution de la localisation des appareils IoT
- Solveurs de localisation et charge utile de l'appareil

Résolution de la localisation des appareils IoT

Utilisez la localisation de l' AWS IoT Core appareil pour décoder les données de mesure de vos appareils et déterminez l'emplacement de l'appareil à l'aide de solveurs tiers. L'emplacement résolu est généré sous forme de charge utile GeoJSON avec les coordonnées géographiques et les informations de précision. Vous pouvez déterminer l'emplacement de votre appareil à partir de la AWS IoT console, de l' AWS IoT Wireless API ou AWS CLI.

Rubriques

- Résolution de la localisation de l'appareil (console)
- Résolution de la localisation de l'appareil (API)

- Résolution des erreurs lors de la résolution de l'emplacement

Résolution de la localisation de l'appareil (console)

Pour résoudre l'emplacement de l'appareil (console)

1. Accédez à la page de localisation de l'appareil dans la AWS IoT console.
2. Obtenez les données de mesure de la charge utile à partir des journaux de votre appareil ou CloudWatch des journaux, et saisissez-les dans la section Position de résolution via la charge utile.

   Le code suivant montre un exemple de charge utile JSON. La charge utile contient des données de mesure cellulaires et Wi-Fi. Si votre charge utile contient d'autres types de données de mesure, le solveur le plus précis sera utilisé. Pour plus d'informations et des exemples de charges utiles, consultez the section called "Solveurs de localisation et charge utile de l'appareil".

   > **Note**
   >
   > La charge utile JSON doit contenir au moins un type de données de mesure.

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress": "54.240.198.35"
    },
    "WiFiAccessPoints": [{
        "MacAddress": "A0:EC:F9:1E:32:C1",
        "Rss": -77
    }],
    "CellTowers": {
        "Gsm": [{
            "Mcc": 262,
            "Mnc": 1,
            "Lac": 5126,
            "GeranCid": 16504,
            "GsmLocalId": {
                "Bsic": 6,
                "Bcch": 82
            },
```

```
            "GsmTimingAdvance": 1,
            "RxLevel": -110,
            "GsmNmr": [{
                "Bsic": 7,
                "Bcch": 85,
                "RxLevel": -100,
                "GlobalIdentity": {
                    "Lac": 1,
                    "GeranCid": 1
                }
            }]
        }],
        "Wcdma": [{
            "Mcc": 262,
            "Mnc": 7,
            "Lac": 65535,
            "UtranCid": 14674663,
            "WcdmaNmr": [{
                    "Uarfcndl": 10786,
                    "UtranCid": 14674663,
                    "Psc": 149
                },
                {
                    "Uarfcndl": 10762,
                    "UtranCid": 14674663,
                    "Psc": 211
                }
            ]
        }],
        "Lte": [{
            "Mcc": 262,
            "Mnc": 2,
            "EutranCid": 2898945,
            "Rsrp": -50,
            "Rsrq": -5,
            "LteNmr": [{
                    "Earfcn": 6300,
                    "Pci": 237,
                    "Rsrp": -60,
                    "Rsrq": -6,
                    "EutranCid": 2898945
                },
                {
                    "Earfcn": 6300,
```

```
                    "Pci": 442,
                    "Rsrp": -70,
                    "Rsrq": -7,
                    "EutranCid": 2898945
                }
            ]
        }]
    }
}
```

3. Pour résoudre les informations de localisation, choisissez Résoudre.

   Les informations de localisation sont de type blob et renvoyées sous forme de charge utile utilisant le format GeoJSON, qui est un format utilisé pour coder les structures de données géographiques. La charge utile contient :

   - Les WGS84 coordonnées géographiques, qui incluent les informations de latitude et de longitude. Il peut également inclure une information d'altitude.
   - Type d'informations de localisation signalées, telles que Point. Un type d'emplacement de point représente l'emplacement sous forme de WGS84 latitude et de longitude, codé sous forme de point GeoJSON.
   - Les informations de précision horizontale et verticale, qui indiquent la différence, en mètres, entre les informations de localisation estimées par les solveurs et l'emplacement réel de l'appareil.
   - Le niveau de confiance, qui indique l'incertitude de la réponse à l'estimation de l'emplacement. La valeur par défaut est 0,68, ce qui indique une probabilité de 68 % que l'emplacement réel de l'appareil se situe dans le rayon d'incertitude de l'emplacement estimé.
   - Ville, État, pays et code postal où se trouve l'appareil. Ces informations ne seront communiquées que lorsque le solveur de recherche inversée IP est utilisé.
   - Les informations d'horodatage, qui correspondent à la date et à l'heure auxquelles la localisation a été résolue. Il utilise le format d'horodatage Unix.

   Le code suivant montre un exemple de charge utile GeoJSON renvoyée en résolvant l'emplacement.

> **Note**
>
> Si AWS IoT Core l'emplacement de l'appareil signale des erreurs lors de la tentative de résolution de la localisation, vous pouvez résoudre les erreurs et résoudre la localisation. Pour de plus amples informations, veuillez consulter Résolution des erreurs lors de la résolution de l'emplacement.


```
{
    "coordinates": [
        13.376076698303223,
        52.51823043823242
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 45,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 303,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T12:23:58.189Z"
    }
}
```


4. Accédez à la section Emplacement des ressources et vérifiez les informations de géolocalisation indiquées par AWS IoT Core Device Location. Vous pouvez copier la charge utile pour l'utiliser avec d'autres applications et Service AWS s. Par exemple, vous pouvez utiliser Emplacement pour envoyer vos données de localisation géographique à Amazon Location Service.

Résolution de la localisation de l'appareil (API)

Pour déterminer l'emplacement de l'appareil à l'aide de l' AWS IoT Wireless API, utilisez l'opération GetPositionEstimateAPI ou la commande get-position-estimateCLI. Spécifiez les données de mesure de la charge utile en entrée et exécutez l'opération API pour déterminer l'emplacement de l'appareil.

> **Note**
>
> L'opération `GetPositionEstimate` API ne stocke aucune information sur l'appareil ou l'état et ne peut pas être utilisée pour récupérer des données de localisation historiques. Il effectue une opération unique qui résout les données de mesure et produit l'emplacement estimé. Pour récupérer les informations de localisation, vous devez spécifier les informations de charge utile chaque fois que vous effectuez cette opération d'API.

La commande suivante montre un exemple de la marche à suivre pour résoudre l'emplacement à l'aide de cette opération d'API.

> **Note**
>
> Lorsque vous exécutez la commande CLI `get-position-estimate`, vous devez spécifier le fichier JSON de sortie comme première entrée. Ce fichier JSON stockera les informations de localisation estimées obtenues en réponse à la CLI au format GeoJSON. Par exemple, la commande suivante enregistre les informations de localisation dans le *`locationout.json`* fichier.

```
aws iotwireless get-position-estimate locationout.json \
    --ip IpAddress=""54.240.198.35"" \
    --wi-fi-access-points \
        MacAddress="A0:EC:F9:1E:32:C1",Rss=-75 \
        MacAddress="A0:EC:F9:15:72:5E",Rss=-67
```

Cet exemple inclut à la fois les points d'accès Wi-Fi et l'adresse IP comme types de mesure. AWS IoT Core L'emplacement de l'appareil choisit entre le solveur Wi-Fi et le solveur de recherche inversée IP, et le solveur est sélectionné avec la plus grande précision.

L'emplacement résolu est renvoyé sous forme de charge utile qui utilise le format GeoJSON, qui est un format utilisé pour coder les structures de données géographiques. Il est ensuite enregistré dans le *`locationout.json`* fichier. La charge utile contient les coordonnées de WGS84 latitude et de longitude, les informations de précision et de niveau de confiance, le type de données de localisation et l'horodatage auquel la position a été résolue.

```
{
```

```
    "coordinates": [
        13.37704086303711,
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z"
    }
}
```

Résolution des erreurs lors de la résolution de l'emplacement

Lorsque vous tentez de résoudre l'emplacement, l'un des codes d'erreur suivants peut s'afficher. AWS IoT Core L'emplacement du périphérique peut générer une erreur lors de l'utilisation de l'`GetPositionEstimate`API, ou bien faire référence au numéro de ligne correspondant à l'erreur dans la AWS IoT console.

- Erreur 400

  Cette erreur indique que le format de la charge utile de l'appareil JSON ne peut pas être validé par AWS IoT Core Device Location. L'erreur peut se produire pour les raisons suivantes :

  - Les données de mesure JSON ne sont pas formatées correctement.
  - La charge utile contient uniquement les informations d'horodatage.
  - Les paramètres des données de mesure, tels que l'adresse IP, ne sont pas valides.

  Pour résoudre cette erreur, vérifiez si votre fichier JSON est correctement formaté et s'il contient des données provenant d'un ou de plusieurs types de mesures en entrée. Si l'adresse IP n'est pas valide, pour plus d'informations sur la manière dont vous pouvez fournir une adresse IP valide pour résoudre l'erreur, consultez Solveur de recherche inversée IP.

- Erreur 403

Cette erreur indique que vous n'êtes pas autorisé à effectuer l'opération d'API ou à utiliser la AWS IoT console pour récupérer l'emplacement de l'appareil. Pour résoudre cette erreur, vérifiez que vous disposez des autorisations requises pour effectuer cette action. Cette erreur peut se produire si votre AWS Management Console session ou votre jeton de AWS CLI session a expiré. Pour résoudre cette erreur, actualisez le jeton de session pour l'utiliser AWS CLI, ou déconnectez-vous du, AWS Management Console puis connectez-vous à l'aide de vos informations d'identification.

- Erreur 404

  Cette erreur indique qu'aucune information de localisation n'a été trouvée ou résolue par AWS IoT Core Device Location. L'erreur peut être due à des cas tels que l'insuffisance des données lors de la saisie des données de mesure. Par exemple :

  - L'adresse MAC ou les informations de la tour cellulaire ne sont pas complètes.
  - L'adresse IP n'est pas disponible pour rechercher et récupérer l'emplacement.
  - La charge utile du GNSS n'est pas suffisante.

  Pour résoudre l'erreur dans de tels cas, vérifiez si vos données de mesure contiennent suffisamment d'informations nécessaires pour localiser l'appareil.

- Erreur 500

  Cette erreur indique qu'une exception interne au serveur s'est produite lorsque AWS IoT Core Device Location a tenté de résoudre l'emplacement. Pour tenter de corriger cette erreur, actualisez la session et réessayez d'envoyer les données de mesure à résoudre.

Résolution de la localisation de l'appareil à l'aide des rubriques MQTT AWS IoT Core relatives à la localisation

Vous pouvez utiliser des sujets MQTT réservés pour obtenir les dernières informations de localisation de vos appareils grâce à la fonction de localisation des AWS IoT Core appareils.

Format des rubriques MQTT relatives à la localisation des appareils

Les rubriques réservées pour la localisation des AWS IoT Core appareils utilisent le préfixe suivant :

```
$aws/device_location/{customer_device_id}/
```

Pour créer une rubrique complète, remplacez *customer_device_id* par votre identifiant unique que vous utilisez pour identifier votre appareil. Nous vous recommandons de spécifier le`WirelessDeviceId`, par exemple pour les appareils LoRa WAN et Sidewalk*thingName*, et si votre appareil est enregistré en tant qu' AWS IoT objet. Vous ajoutez ensuite la rubrique avec le stub de rubrique, tel que `get_position_estimate` ou `get_position_estimate/accepted` comme indiqué dans la section suivante.

> **Note**
>
> Le *{customer_device_id}* ne peut contenir que des lettres, des chiffres et des tirets. Lorsque vous vous abonnez à des rubriques relatives à la localisation des appareils, vous ne pouvez utiliser le signe plus (+) que comme caractère générique. Par exemple, vous pouvez utiliser le caractère + générique pour *{customer_device_id}* pour obtenir les informations de localisation de vos appareils. Lorsque vous vous abonnez à la rubrique `$aws/device_location/+/get_position_estimate/accepted`, un message contenant les informations de localisation des appareils correspondant à n'importe quel identifiant d'appareil est publié si le problème a été résolu avec succès.

Les rubriques réservées utilisées pour interagir avec la localisation de l' AWS IoT Core appareil sont les suivantes.

Format des rubriques MQTT de Device location

Rubrique	Opérations autorisées	Description
$aws/device_location/ / get_position_estimate *customer_device_id*	Publication	Un appareil publie dans cette rubrique pour obtenir les données de mesure brutes numérisées à résoudre en fonction de l'emplacement de AWS IoT Core l'appareil.
$aws/device_location/ / get_position_estimate/ accepted *customer_ device_id*	S’abonner	AWS IoT Core La localisation de l'appareil publie les informations de localisation dans cette rubrique lorsqu'elle résout avec succès la localisation de l'appareil.
$aws/device_locati on/ /get_position_esti	S’abonner	AWS IoT Core L'emplacement du périphéri que publie les informations d'erreur dans

Rubrique	Opérations autorisées	Description
mate/rejeté *customer_device_id*		cette rubrique lorsqu'il ne parvient pas à résoudre l'emplacement du périphérique.

Rubriques MQTT relatives à la politique de localisation des appareils

Pour recevoir des messages provenant de sujets relatifs à la localisation de l'appareil, celui-ci doit utiliser une politique lui permettant de se connecter à la passerelle de l' AWS IoT appareil et de s'abonner aux sujets MQTT.

Voici un exemple de la politique requise pour recevoir des messages pour les différentes rubriques.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/
device_location/customer_device_id/get_position_estimate"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/
device_location/customer_device_id/get_position_estimate/accepted",
        "arn:aws:iot:region:account:topic/$aws/
device_location/customer_device_id/get_position_estimate/rejected"
      ]
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/
device_location/customer_device_id/get_position_estimate/accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/
device_location/customer_device_id/get_position_estimate/rejected"
      ]
    }
  ]
}
```

Sujets relatifs à la localisation des appareils et charge utile

Vous trouverez ci-dessous les rubriques relatives à la localisation des AWS IoT Core appareils, le format de la charge utile des messages et un exemple de politique pour chaque rubrique.

Rubriques

- /get_position_estimate
- /get_position_estimate/accepted
- /get_position_estimate/rejeté

/get_position_estimate

Publiez un message dans cette rubrique pour obtenir les données de mesure brutes de l'appareil à résoudre en fonction de l'emplacement de AWS IoT Core l'appareil.

```
$aws/device_location/customer_device_id/get_position_estimate
```

AWS IoT Core Device Location répond en publiant sur l'un /get_position_estimate/accepted ou l'autre/get_position_estimate/rejeté.

> **Note**
>
> Le message publié dans cette rubrique doit être une charge utile JSON valide. Si le message d'entrée n'est pas au format JSON valide, vous n'obtiendrez aucune réponse. Pour plus d'informations, consultez Message payload. (Charge utile du message)

Charge utile du message

Le format de charge utile du message suit une structure similaire à celle du corps de la demande AWS IoT Wireless d'opération d'API, `GetPositionEstimate`. Il contient :

- Chaîne facultative `Timestamp`, qui correspond à la date et à l'heure de résolution de l'emplacement. La chaîne `Timestamp` peut avoir une longueur minimale de 1 et une longueur maximale de 10.
- Chaîne facultative `MessageId`, qui peut être utilisée pour mapper la demande à la réponse. Si vous spécifiez cette chaîne, le message publié dans les rubriques `get_position_estimate/accepted` ou `get_position_estimate/rejected` contiendra ce `MessageId`. La chaîne `MessageID` peut avoir une longueur minimale de 1 et une longueur maximale de 256.
- Les données de mesure de l'appareil qui contient un ou plusieurs des types de mesure suivants :
  - `WiFiAccessPoint`
  - `CellTowers`
  - `IpAddress`
  - `Gnss`

Ce qui suit montre un exemple de charge utile de message.

```
{
    "Timestamp": "1664313161",
    "MessageId": "ABCD1",
    "WiFiAccessPoints":  [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",
            "Rss": -66
        }
    ],
    "Ip":{
        "IpAddress": "54.192.168.0"
```

```
    },
    "Gnss":{
      "Payload":"8295A614A2029517F4F77C0A7823B161A6FC57E25183D96535E3689783F6CA48",
      "CaptureTime":1354393948
    }
}
```

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/
device_location/customer_device_id/get_position_estimate"
      ]
    }
  ]
}
```

/get_position_estimate/accepted

AWS IoT Core L'emplacement de l'appareil publie une réponse à cette rubrique lorsque vous renvoyez les informations de localisation résolues pour votre appareil. Les informations de localisation sont renvoyées en format GeoJSON.

```
$aws/device_location/customer_device_id/get_position_estimate/accepted
```

Vous trouverez ci-dessous la charge utile des messages et un exemple de politique.

Charge utile du message

Voici un exemple de la charge utile du message au format GeoJSON. Si vous avez spécifié un `MessageId` dans vos données de mesure brutes et que AWS IoT Core Device Location a résolu les informations de localisation avec succès, la charge utile du message renvoie les mêmes `MessageId` informations.

```
{
    "coordinates": [
        13.37704086303711,
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "country": "USA",
        "state": "CA",
        "city": "Sunnyvalue",
        "postalCode": "91234",
        "timestamp": "2022-11-18T14:03:57.391Z",
        "messageId": "ABCD1"
    }
}
```

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
```

```
        "arn:aws:iot:region:account:topicfilter/$aws/
device_location/customer_device_id/get_position_estimate/accepted"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/
device_location/customer_device_id/get_position_estimate/accepted"
      ]
    }
  ]
}
```

/get_position_estimate/rejeté

AWS IoT Core L'emplacement de l'appareil publie une réponse d'erreur à cette rubrique lorsqu'il ne parvient pas à résoudre l'emplacement de l'appareil.

```
$aws/device_location/customer_device_id/get_position_estimate/rejected
```

Vous trouverez ci-dessous la charge utile des messages et exemple de politique. Pour plus d'informations sur les erreurs, consultez Résolution des erreurs lors de la résolution de l'emplacement.

Charge utile du message

Voici un exemple de charge utile du message qui fournit le code d'erreur et le message, qui indiquent pourquoi AWS IoT Core Device Location n'a pas réussi à résoudre les informations de localisation. Si vous avez spécifié un `MessageId` lorsque vous avez fourni vos données de mesure brutes et que l'emplacement de l' AWS IoT Core appareil n'a pas résolu les informations de localisation, les mêmes `MessageId` informations seront renvoyées dans la charge utile du message.

```
{
    "errorCode": 500,
    "errorMessage":"Internal server error",
    "messageId": "ABCD1"
```

```
}
```

Exemple de stratégie

Voici un exemple de document de stratégie requise :

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/
device_location/customer_device_id/get_position_estimate/rejected"
      ]
    },
    {
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/
device_location/customer_device_id/get_position_estimate/rejected"
      ]
    }
  ]
}
```

Solveurs de localisation et charge utile de l'appareil

Les solveurs de localisation sont des algorithmes qui peuvent être utilisés pour déterminer la position de vos appareils IoT. AWS IoT Core Device Location prend en charge les solveurs de localisation suivants. Vous verrez des exemples du format de charge utile JSON pour ces types de mesures, les appareils pris en charge par le solveur et la manière dont la localisation est résolue.

Pour déterminer l'emplacement de l'appareil, spécifiez un ou plusieurs de ces types de données de mesure. Une position unique et résolue sera renvoyée pour toutes les données de mesure combinées.

Rubriques

- Solveur basé sur le Wi-Fi
- Solveur cellulaire
- Solveur de recherche inversée IP
- solveur GNSS

Solveur basé sur le Wi-Fi

Utilisez le solveur basé sur le Wi-Fi pour déterminer la position à l'aide des informations de numérisation provenant des points d'accès Wi-Fi. Le solveur prend en charge la technologie WLAN et peut être utilisé pour calculer l'emplacement des appareils IoT généraux et des appareils sans fil LoRa WAN.

Les appareils LoRa WAN doivent être équipés du chipset LoRa Edge, qui peut décoder les informations de numérisation Wi-Fi entrantes. LoRa Edge est une plate-forme à très faible consommation qui intègre un LoRa émetteur-récepteur longue portée, un scanner GNSS multi-constellations et un scanner MAC Wi-Fi passif ciblant les applications de géolocalisation. Lorsqu'un message de liaison montante est reçu de l'appareil, les données de numérisation Wi-Fi sont envoyées à l'emplacement de l' AWS IoT Core appareil, et l'emplacement est estimé sur la base des résultats du scan Wi-Fi. Les informations décodées sont ensuite transmises au solveur basé sur le Wi-Fi pour récupérer les informations de localisation.

Exemple de charge utile d'un solveur basé sur le Wi-Fi

Le code suivant montre un exemple de la charge utile JSON provenant de l'appareil qui contient les données de mesure. Lorsque AWS IoT Core Device Location reçoit ces données en entrée, il envoie une requête HTTP au fournisseur du solveur pour résoudre les informations de localisation. Pour récupérer les informations, spécifiez les valeurs de l'adresse MAC et du flux RSS (intensité du signal reçu). Pour ce faire, fournissez la charge utile JSON en utilisant ce format ou utilisez le paramètre WiFiAccessPointsobject de l'opération d'GetPositionEstimateAPI.

```
{
    "Timestamp": 1664313161,    // optional
    "WiFiAccessPoints": [
```

```
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",  // required
            "Rss": -75                          // required
        }
    ]
}
```

Solveur cellulaire

Vous pouvez utiliser le solveur cellulaire pour déterminer l'emplacement à l'aide des données de mesure obtenues à partir des pylônes de téléphonie cellulaire. Le solveur prend en charge les technologies suivantes. Une seule information de localisation résolue est obtenue, même si vous incluez des données de mesure issues de l'une ou de l'ensemble de ces technologies.

- GSM
- CDMA
- WCDMA
- TD-SCDMA
- LTE

Exemples de charge utile d'un solveur cellulaire

Le code suivant montre des exemples de charge utile JSON de l'appareil qui contient des données de mesure cellulaire. Lorsque AWS IoT Core Device Location reçoit ces données en entrée, il envoie une requête HTTP au fournisseur du solveur pour résoudre les informations de localisation. Pour récupérer les informations, vous devez soit fournir la charge utile JSON en utilisant ce format dans la console, soit spécifier des valeurs pour le CellTowersparamètre de l'opération d'GetPositionEstimateAPI. Vous pouvez fournir les données de mesure en spécifiant les valeurs des paramètres à l'aide de l'une ou de l'ensemble de ces technologies cellulaires.

LTE (évolution à long terme)

Lorsque vous utilisez ces données de mesure, vous devez spécifier des informations telles que le réseau et le code du pays du réseau mobile, ainsi que des paramètres supplémentaires facultatifs, notamment des informations sur l'identifiant local. Le code suivant montre un exemple du format de charge utile. Pour plus d'informations sur ces paramètres, consultez LTE object. (Objet LTE)

```
{
```

```
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Lte": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "EutranCid": int,             // required. Make sure that you use int for
 EutranCid.
          "Tac": int,                   // optional
          "LteLocalId": {               // optional
              "Pci": int,               // required
              "Earfcn": int,            // required
          },
          "LteTimingAdvance": int,      // optional
          "Rsrp": int,                  // optional
          "Rsrq": float,                // optional
          "NrCapable": boolean,         // optional
          "LteNmr": [                   // optional
                {
                    "Pci": int,         // required
                    "Earfcn": int,      // required
                    "EutranCid": int,   // required
                    "Rsrp": int,        // optional
                    "Rsrq": float       // optional
                }
            ]
         }
      ]
   }
}
```

GSM (Système mondial de communications mobiles)

Lorsque vous utilisez ces données de mesure, vous devez spécifier des informations telles que le réseau et le code pays du réseau mobile, les informations de la station de base et des paramètres supplémentaires facultatifs. Le code suivant montre un exemple du format de charge utile. Pour plus d'informations sur ces paramètres, consultez GSM object. (Objet GSM)

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Gsm": [
        {
```

```
            "Mcc": int,                    // required
            "Mnc": int,                    // required
            "Lac": int,                    // required
            "GeranCid": int,               // required
            "GsmLocalId": {                // optional
                "Bsic": int,               // required
                "Bcch": int,               // required
            },
            "GsmTimingAdvance": int,       // optional
            "RxLevel": int,                // optional
            "GsmNmr": [                    // optional
              {
                  "Bsic": int,             // required
                  "Bcch": int,             // required
                  "RxLevel": int,          // optional
                  "GlobalIdentity": {
                      "Lac": int,          // required
                      "GeranCid": int      // required
                  }
               }
            ]
        }
    ]
}
```

CDMA (accès multiple par division de code)

Lorsque vous utilisez ces données de mesure, vous devez spécifier des informations telles que la puissance du signal et les informations d'identification, les informations de la station de base et des paramètres supplémentaires facultatifs. Le code suivant montre un exemple du format de charge utile. Pour plus d'informations sur ces paramètres, consultez CDMA object. (Objet CDMA)

```
{
    "Timestamp": 1664313161,               // optional
    "CellTowers": {
        "Cdma": [
        {
            "SystemId": int,                // required
            "NetworkId": int,               // required
            "BaseStationId": int,           // required
            "RegistrationZone": int,        // optional
            "CdmaLocalId": {                // optional
              "PnOffset": int,              // required
```

```
              "CdmaChannel": int,           // required
            },
            "PilotPower": int,              // optional
            "BaseLat": float,               // optional
            "BaseLng": float,               // optional
            "CdmaNmr": [                    // optional
                {
                    "PnOffset": int,        // required
                    "CdmaChannel": int,     // required
                    "PilotPower": int,      // optional
                    "BaseStationId": int    // optional
                }
             ]
          }
       ]
    }
}
```

WCDMA (Wideband code-division multiple access)/ (accès multiple par répartition en code à large bande)

Lorsque vous utilisez ces données de mesure, vous devez spécifier des informations telles que le code du réseau et du pays, la puissance du signal et les informations d'identification, les informations de la station de base et des paramètres supplémentaires facultatifs. Le code suivant montre un exemple du format de charge utile. Pour plus d'informations sur ces paramètres, consultez CDMA object. (Objet CDMA)

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Wcdma": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "UtranCid": int,              // required
          "Lac": int,                   // optional
          "WcdmaLocalId": {             // optional
              "Uarfcndl": int,          // required
              "Psc": int,               // required
          },
          "Rscp": int,                  // optional
          "Pathloss": int,              // optional
          "WcdmaNmr": [                 // optional
```

```
                {
                  "Uarfcndl": int,      // required
                  "Psc": int,           // required
                  "UtranCid": int,      // required
                  "Rscp": int,          // optional
                  "Pathloss": int,      // optional
                }
            ]
        }
    ]
  }
}
```

TD-SCDMA (Time division synchronous code-division multiple access)/ (accès multiple synchrone par répartition dans le temps par code)

Lorsque vous utilisez ces données de mesure, vous devez spécifier des informations telles que le code du réseau et du pays, la puissance du signal et les informations d'identification, les informations de la station de base et des paramètres supplémentaires facultatifs. Le code suivant montre un exemple du format de charge utile. Pour plus d'informations sur ces paramètres, consultez CDMA object. (Objet CDMA)

```
{
    "Timestamp": 1664313161,          // optional
    "CellTowers": {
        "Tdscdma": [
        {
          "Mcc": int,                  // required
          "Mnc": int,                  // required
          "UtranCid": int,             // required
          "Lac": int,                  // optional
          "TdscdmaLocalId": {          // optional
              "Uarfcn": int,           // required
              "CellParams": int,       // required
          },
          "TdscdmaTimingAdvance": int,  // optional
          "Rscp": int,                 // optional
          "Pathloss": int,             // optional
          "TdscdmaNmr": [              // optional
                {
                  "Uarfcn": int,       // required
                  "CellParams": int,   // required
                  "UtranCid": int,     // optional
```

```
                "Rscp": int,          // optional
                "Pathloss": int,      // optional
              }
            ]
        }
    ]
  }
}
```

Solveur de recherche inversée IP

Vous pouvez utiliser le solveur de recherche inversée IP pour déterminer l'emplacement en utilisant l'adresse IP comme entrée. Le solveur peut obtenir les informations de localisation à partir des appareils qui ont été approvisionnés. AWS IoT Spécifiez les informations d'adresse IP en utilisant un format qui est IPv4 soit le modèle IPv6 standard, soit le modèle compressé IPv6 hexadécimal. Vous obtenez ensuite l'estimation de localisation résolue, y compris des informations supplémentaires telles que la ville et le pays où se trouve l'appareil.

> **Note**
>
> En utilisant la recherche inversée d'adresse IP, vous acceptez de ne pas l'utiliser dans le but d'identifier ou de localiser une adresse résidentielle ou municipale spécifique.

Exemple de charge utile d'un solveur de recherche inversée IP

Le code suivant montre un exemple de la charge utile JSON provenant de l'appareil qui contient les données de mesure. Lorsque AWS IoT Core Device Location reçoit les informations d'adresse IP contenues dans les données de mesure, il recherche ces informations dans la base de données du fournisseur du solveur, qui est ensuite utilisée pour résoudre les informations de localisation. Pour récupérer les informations, fournissez la charge utile JSON en utilisant ce format ou spécifiez des valeurs pour le paramètre Ip de l'opération d'GetPositionEstimateAPI.

> **Note**
>
> Lorsque ce solveur est utilisé, la ville, l'État, le pays et le code postal où se trouve l'appareil sont également indiqués en plus des coordonnées. Pour obtenir un exemple, consultez Résolution de la localisation de l'appareil (console).

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress":"54.240.198.35"
      }
}
```

solveur GNSS

Utilisez le solveur GNSS (Global Navigation Satellite System) pour récupérer l'emplacement de l'appareil à l'aide des informations contenues dans les messages de résultats du scan GNSS ou les messages NAV. Vous pouvez éventuellement fournir des informations d'assistance GNSS supplémentaires, ce qui réduit le nombre de variables que le solveur doit utiliser pour rechercher des signaux. En fournissant ces informations d'assistance, qui incluent la position, l'altitude, le temps de capture et les informations de précision, le solveur peut facilement identifier les satellites en vue et calculer la position de l'appareil.

Ce solveur peut être utilisé avec des périphériques LoRa WAN et d'autres appareils qui ont été approvisionnés avec. AWS IoT Pour les appareils IoT généraux, si les appareils prennent en charge l'estimation de position à l'aide du GNSS, lorsque les informations de numérisation GNSS sont reçues de l'appareil, les émetteurs-récepteurs résolvent les informations de localisation. Pour les appareils LoRa WAN, ils doivent être équipés du chipset LoRa Edge. Lorsqu'un message en liaison montante est reçu du dispositif, les données de numérisation GNSS sont envoyées à AWS IoT Core for LoRaWAN, et l'emplacement est estimé sur la base des résultats de numérisation provenant des émetteurs-récepteurs.

Exemple de charge utile d'un solveur GNSS

Le code suivant montre un exemple de la charge utile JSON provenant de l'appareil qui contient les données de mesure. Lorsque AWS IoT Core Device Location reçoit les informations de numérisation GNSS contenant la charge utile contenue dans les données de mesure, il utilise les émetteurs-récepteurs et toute information d'assistance supplémentaire incluse pour rechercher des signaux et résoudre les informations de localisation. Pour récupérer les informations, fournissez la charge utile JSON en utilisant ce format ou spécifiez des valeurs pour le paramètre Gnss de l'opération d'GetPositionEstimateAPI.

> **Note**
>
> Avant que AWS IoT Core l'emplacement du périphérique puisse résoudre l'emplacement du périphérique, vous devez supprimer l'octet de destination de la charge utile.

```
{
    "Timestamp": 1664313161,                    // optional
    "Gnss": {
        "AssistAltitude": number,               // optional
        "AssistPosition": [ number ],           // optional
        "CaptureTime": number,                  // optional
        "CaptureTimeAccuracy": number,          // optional
        "Payload": "string",                    // required
        "Use2DSolver": boolean                  // optional
   }
}
```

Messages d'événements

Cette section contient des informations sur les messages publiés AWS IoT lorsque des objets ou des tâches sont mis à jour ou modifiés. Pour plus d'informations sur le AWS IoT Events service qui vous permet de créer des détecteurs pour surveiller les défaillances ou les changements de fonctionnement de vos appareils, et pour déclencher des actions lorsqu'ils se produisent, voir AWS IoT Events.

Comment les messages d'événement sont générés

AWS IoT publie des messages d'événements lorsque certains événements se produisent. Par exemple, le registre génère des événements quand des objets sont ajoutés, mis à jour ou supprimés. Chaque événement génère l'envoi d'un seul message. Les messages d'événements sont publiés sur MQTT avec une charge utile JSON. Le contenu de la charge utile dépend du type d'événement.

> **Note**
>
> Il est garanti que les messages d'événements sont publiés une fois. Ils peuvent être publiés plusieurs fois. L'ordre des messages d'événement n'est pas garanti.

Politique de réception des messages d'événement

Pour recevoir des messages d'événements, votre appareil doit utiliser une politique appropriée lui permettant de se connecter à la passerelle de l' AWS IoT appareil et de s'abonner aux sujets d'événements MQTT. Vous devez aussi vous abonner aux filtres de rubriques appropriés.

Voici un exemple de stratégie requise pour recevoir des événements de cycle de vie :

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "iot:Subscribe",
            "iot:Receive"
```

```
        ],
        "Resource":[
            "arn:aws:iot:region:account:/$aws/events/*"
        ]
    }]
}
```

Activez les événements pour AWS IoT

Avant que les abonnés aux rubriques réservées puissent recevoir des messages, vous devez activer les messages d'événement depuis AWS Management Console ou à l'aide de l'API ou de la CLI. Pour plus d'informations sur les messages d'événements gérés par les différentes options, consultez le Tableau des paramètres de configuration des AWS IoT événements.

- Pour activer les messages d'événements, accédez à l'onglet Paramètres de la AWS IoT console, puis, dans la section Messages basés sur les événements, choisissez Gérer les événements. Vous pouvez spécifier les événements que vous souhaitez gérer.
- Pour contrôler les types d'événements publiés à l'aide de l'API ou de la CLI, appelez l'UpdateEventConfigurationsAPI ou utilisez la commande update-event-configurations CLI. Par exemple :

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\":
 true}}"
```

> **Note**
>
> L'échappement de tous les guillemets (") est effectué avec des barres obliques inverses (\).

Vous pouvez obtenir la configuration actuelle de l'événement en appelant l'DescribeEventConfigurationsAPI ou en utilisant la commande describe-event-configurations CLI. Par exemple :

```
aws iot describe-event-configurations
```

Tableau des paramètres AWS IoT de configuration des événements

Catégorie d'événement (AWS IoT Console : Paramètres : messages basés sur des événements)	Valeur de la clé **`eventConfigurations`** (AWS CLI/API)	Rubrique du message d'événement
(Ne peut être configuré qu'à l'aide de AWS CLI/API)	`CA_CERTIFICATE`	`$aws/events/certificates/registered/` *`caCertificateId`*
(Ne peut être configuré qu'à l'aide de AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/connected/` *`clientId`*
(Ne peut être configuré qu'à l'aide de AWS CLI/API)	`CERTIFICATE`	`$aws/events/presence/disconnected/` *`clientId`*
(Ne peut être configuré qu'à l'aide de AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/subscribed/` *`clientId`*
(Ne peut être configuré qu'à l'aide de AWS CLI/API)	`CERTIFICATE`	`$aws/events/subscriptions/unsubscribed/` *`clientId`*
Tâche terminée, annulée	`JOB`	`$aws/events/job/`*`jobID`*`/canceled`
Tâche terminée, annulée	`JOB`	`$aws/events/job/`*`jobID`*`/cancellation_in_progress`
Tâche terminée, annulée	`JOB`	`$aws/events/job/`*`jobID`*`/completed`
Tâche terminée, annulée	`JOB`	`$aws/events/job/`*`jobID`*`/deleted`

Catégorie d'événement (AWS IoT Console : Paramètres : messages basés sur des événements)	Valeur de la clé **`eventConfigurations`** (AWS CLI/API)	Rubrique du message d'événement
Tâche terminée, annulée	`JOB`	`$aws/events/job/`*`jobID`*`/deletion_in_progress`
Exécution de tâche : réussite, échec, rejet, annulation, suppression	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/canceled`
Exécution de tâche : réussite, échec, rejet, annulation, suppression	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/deleted`
Exécution de tâche : réussite, échec, rejet, annulation, suppression	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/failed`
Exécution de tâche : réussite, échec, rejet, annulation, suppression	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/rejected`
Exécution de tâche : réussite, échec, rejet, annulation, suppression	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/removed`
Exécution de tâche : réussite, échec, rejet, annulation, suppression	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/succeeded`
Exécution de tâche : réussite, échec, rejet, annulation, suppression	`JOB_EXECUTION`	`$aws/events/jobExecution/ `*`jobID`*`/timed_out`

Catégorie d'événement (AWS IoT Console : Paramètres : messages basés sur des événements)	Valeur de la clé **`eventConfigurations`** (AWS CLI/API)	Rubrique du message d'événement
Objet : créé, mis à jour, supprimé	`THING`	`$aws/events/thing/` *`thingName`* `/created`
Objet : créé, mis à jour, supprimé	`THING`	`$aws/events/thing/` *`thingName`* `/updated`
Objet : créé, mis à jour, supprimé	`THING`	`$aws/events/thing/` *`thingName`* `/deleted`
Groupe d'objets : ajouté, supprimé	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/created`
Groupe d'objets : ajouté, supprimé	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/updated`
Groupe d'objets : ajouté, supprimé	`THING_GROUP`	`$aws/events/thingGroup/` *`thingGroupName`* `/deleted`
Hiérarchie des groupes d'objets : ajouté, supprimé	`THING_GROUP_HIERARCHY`	`$aws/events/thingGroupHierarchy/thingGroup/` *`parentThingGroupName`* `/childThingGroup/` *`childThingGroupName`* `/added`

Catégorie d'événement (AWS IoT Console : Paramètres : messages basés sur des événements)	Valeur de la clé **`eventConfigurations`** (AWS CLI/API)	Rubrique du message d'événement
Hiérarchie des groupes d'objets : ajouté, supprimé	`THING_GROUP_HIERARCHY`	`$aws/events/thingGroupHierarchy/thingGroup/` *`parentThingGroupName`* `/childThingGroup/` *`childThingGroupName`* `/removed`
Appartenance à un groupe d'objets : ajouté, supprimé	`THING_GROUP_MEMBERSHIP`	`$aws/events/thingGroupMembership/thingGroup/` *`thingGroupName`* `/thing/`*`thingName`* `/added`
Appartenance à un groupe d'objets : ajouté, supprimé	`THING_GROUP_MEMBERSHIP`	`$aws/events/thingGroupMembership/thingGroup/` *`thingGroupName`* `/thing/`*`thingName`* `/removed`
Type d'objet : créé, mis à jour, supprimé	`THING_TYPE`	`$aws/events/thingType/` *`thingTypeName`* `/created`
Type d'objet : créé, mis à jour, supprimé	`THING_TYPE`	`$aws/events/thingType/` *`thingTypeName`* `/updated`
Type d'objet : créé, mis à jour, supprimé	`THING_TYPE`	`$aws/events/thingType/` *`thingTypeName`* `/deleted`

Catégorie d'événement (AWS IoT Console : Paramètres : messages basés sur des événements)	Valeur de la clé **`eventConfigurations`** (AWS CLI/API)	Rubrique du message d'événement

Association de types d'objet : ajouté, supprimé	`THING_TYPE_ASSOCIATION`	`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/added`
		
		`$aws/events/thingTypeAssociation/thing/` *`thingName`* `/thingType/` *`thingTypeName`* `/removed`

Événements de registre

Le registre peut publier des messages d'événement lorsque des objets, des types d'objets et des groupes d'objets sont créés, mis à jour ou supprimés. Ces événements ne sont cependant pas disponibles par défaut. Pour plus d'informations sur la façon d'activer ces événements, consultez Activez les événements pour AWS IoT.

Le registre peut fournir les types d'événements suivants :

- Événements de l’objet
- Événements de types d'objet
- Événements de groupe d'objets

Événements de l’objet

Chose Created/Updated/Deleted

Le registre publie les messages d'événement suivants lorsque des objets sont créés, mis à jour ou supprimés :

- `$aws/events/thing/thingName/created`
- `$aws/events/thing/thingName/updated`
- `$aws/events/thing/thingName/deleted`

Les messages contiennent l'exemple de charge utile suivant :

```
{
    "eventType" : "THING_EVENT",
    "eventId" : "f5ae9b94-8b8e-4d8e-8c8f-b3266dd89853",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 1,
    "thingTypeName" : null,
    "attributes": {
                "attribute3": "value3",
                "attribute1": "value1",
                "attribute2": "value2"
    }
}
```

Les charges utiles contiennent les attributs suivants :

eventType

Défini sur « THING_EVENT ».

eventId

Un ID d'événement unique (chaîne).

timestamp

L'horodatage UNIX du moment où l'événement s'est produit.

fonctionnement

L'opération qui a déclenché l'événement. Les valeurs valides sont :

- CRÉÉ
- MIS À JOUR

- SUPPRIMÉ

accountId

Votre Compte AWS carte d'identité.

thingId

L'ID de l'objet en cours de création, de mise à jour ou de suppression.

thingName

Le nom de l'objet en cours de création, de mise à jour ou de suppression.

versionNumber

La version de l'objet en cours de création, de mise à jour ou de suppression. Cette valeur est définie sur 1 lors de la création d'un objet. Elle augmente de 1 à chaque mise à jour de l'objet.

thingTypeName

Le type d'objet associé à l'objet, le cas échéant. Sinon la valeur est renvoy, `null`.

attributs

Un ensemble de paires nom-valeur associées à l'objet.

Événements de types d'objet

Événements liés au type d'objet :

- Type d'objet Created/Updated/Deprecated/Undeprecated/Deleted
- Type d'objet associé à un objet/dissocié d'un objet

Type d'objet Created/Updated/Deprecated/Undeprecated/Deleted

Le registre publie les messages d'événement suivants lorsque des types d'objets sont créés, mis à jour, déconseillés, déconseillés ou supprimés :

- `$aws/events/thingType/`*`thingTypeName`*`/created`
- `$aws/events/thingType/`*`thingTypeName`*`/updated`
- `$aws/events/thingType/`*`thingTypeName`*`/deleted`

Le message contient l'exemple de charge utile suivant :

```
{
    "eventType" : "THING_TYPE_EVENT",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : false|true,
    "deprecationDate" : null,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "propagatingAttributes": [
        {
     "userPropertyKey": "key",
     "thingAttribute": "model"
        },
        {
     "userPropertyKey": "key",
     "connectionAttribute": "iot:ClientId"
        }
    ],
    "description" : "My thing type"
}
```

Les charges utiles contiennent les attributs suivants :

eventType

Défini sur « THING_TYPE_EVENT ».

eventId

Un ID d'événement unique (chaîne).

timestamp

L'horodatage UNIX du moment où l'événement s'est produit.

fonctionnement

L'opération qui a déclenché l'événement. Les valeurs valides sont :

- CRÉÉ
- MIS À JOUR
- SUPPRIMÉ

accountId

Votre Compte AWS carte d'identité.

thingTypeId

L'ID du type d'objet en cours de création, de mise à jour, de dépréciation ou de suppression.

thingTypeName

Le nom du type d'objet en cours de création, de mise à jour, de dépréciation ou de suppression.

isDeprecated

`true` si le type d'objet est obsolète. Sinon la valeur est renvoy, `false`.

deprecationDate

L'horodatage UNIX associé au moment où ce type d'objet est devenu obsolète.

searchableAttributes

Un ensemble de paires nom-valeur associées au type d'objet qui peut être utilisé pour la recherche.

Attributs de propagation

Liste des attributs de propagation. Un attribut de propagation peut contenir un attribut d'objet, un attribut de connexion et une clé de propriété utilisateur. Pour plus d'informations, consultez la section Ajout d'attributs de propagation pour l'enrichissement des messages.

description

Une description du type d'objet.

Type d'objet associé à un objet/dissocié d'un objet

Le registre publie les messages d'événement suivants lorsqu'un type d'objet est associé à un objet ou dissocié d'un objet.

- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/added`
- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/removed`

Voici un exemple de charge utile `added`. Les charges utiles pour les messages `removed` sont similaires.

```
{
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "eventType" : "THING_TYPE_ASSOCIATION_EVENT",
    "operation" : "ADDED",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName": "myThing",
    "thingTypeName" : "MyThingType",
    "timestamp" : 1234567890123,
}
```

Les charges utiles contiennent les attributs suivants :

eventId

Un ID d'événement unique (chaîne).

eventType

Défini sur « THING_TYPE_ASSOCIATION_EVENT ».

fonctionnement

L'opération qui a déclenché l'événement. Les valeurs valides sont :

- AJOUTÉ
- SUPPRIMÉ

thingId

ID de l'objet dont l'association du type a été modifiée.

thingName

Nom de l'objet dont l'association du type a été modifiée.

thingTypeName

Type d'objet associé à l'objet ou qui n'est plus associé à l'objet.

timestamp

L'horodatage UNIX du moment où l'événement s'est produit.

Événements de groupe d'objets

Événements liés aux groupes d'objets :

- Groupe d'objets Created/Updated/Deleted
- Objet ajouté à un groupe d'objets/retiré d'un groupe d'objets
- Groupe d'objets ajouté à un groupe d'objets/retiré d'un groupe d'objets

Groupe d'objets Created/Updated/Deleted

Le registre publie les messages d'événement suivants lorsqu'un groupe d'objets est créé, mis à jour ou supprimé.

- `$aws/events/thingGroup/`*`groupName`*`/created`
- `$aws/events/thingGroup/`*`groupName`*`/updated`
- `$aws/events/thingGroup/`*`groupName`*`/deleted`

Voici un exemple de charge utile `updated`. Les charges utiles pour `created` et des messages `deleted` sont similaires.

```
{
  "eventType": "THING_GROUP_EVENT",
  "eventId": "8b9ea8626aeaa1e42100f3f32b975899",
  "timestamp": 1603995417409,
  "operation": "UPDATED",
  "accountId": "571EXAMPLE833",
  "thingGroupId": "8757eec8-bb37-4cca-a6fa-403b003d139f",
  "thingGroupName": "Tg_level5",
  "versionNumber": 3,
  "parentGroupName": "Tg_level4",
  "parentGroupId": "5fce366a-7875-4c0e-870b-79d8d1dce119",
  "description": "New description for Tg_level5",
  "rootToParentThingGroups": [
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/TgTopLevel",
      "groupId": "36aa0482-f80d-4e13-9bff-1c0a75c055f6"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level1",
      "groupId": "bc1643e1-5a85-4eac-b45a-92509cbe2a77"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level2",
      "groupId": "0476f3d2-9beb-48bb-ae2c-ea8bd6458158"
```

```
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level3",
      "groupId": "1d9d4ffe-a6b0-48d6-9de6-2e54d1eae78f"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level4",
      "groupId": "5fce366a-7875-4c0e-870b-79d8d1dce119"
    }
  ],
  "attributes": {
    "attribute1": "value1",
    "attribute3": "value3",
    "attribute2": "value2"
  },
  "dynamicGroupMappingId": null
}
```

Les charges utiles contiennent les attributs suivants :

eventType

Défini sur « THING_GROUP_EVENT ».

eventId

Un ID d'événement unique (chaîne).

timestamp

L'horodatage UNIX du moment où l'événement s'est produit.

fonctionnement

L'opération qui a déclenché l'événement. Les valeurs valides sont :

- CRÉÉ
- MIS À JOUR
- SUPPRIMÉ

accountId

Votre Compte AWS carte d'identité.

thingGroupId

L'ID du groupe d'objets en cours de création, de mise à jour ou de suppression.

thingGroupName

Le nom du groupe d'objets en cours de création, de mise à jour ou de suppression.

versionNumber

Version du groupe d'objets. Cette valeur est définie sur 1 lors de la création d'un groupe d'objets. Elle augmente de 1 à chaque mise à jour du groupe d'objets.

parentGroupName

Le nom du groupe d'objets parent (le cas échéant).

parentGroupId

L'ID du groupe d'objets parent (le cas échéant).

description

La description du groupe d'objets.

rootToParentThingGroups

Tableau d'informations sur le groupe d'objets parent. Il existe un élément pour chaque groupe d'objets parent, en commençant par le groupe d'objets racine et en continuant jusqu'au parent du groupe d'objets. Chaque entrée contient les groupes d'objets `groupArn` et `groupId`.

attributs

Un ensemble de paires nom-valeur associées au groupe d'objets.

Objet ajouté à un groupe d'objets/retiré d'un groupe d'objets

Le registre publie les messages d'événement suivants lorsqu'un objet est ajouté à un groupe d'objets ou retiré d'un groupe d'objets.

- `$aws/events/thingGroupMembership/thingGroup/`*`thingGroupName`*`/thing/`*`thingName`*`/added`
- `$aws/events/thingGroupMembership/thingGroup/`*`thingGroupName`*`/thing/`*`thingName`*`/removed`

Les messages contiennent l'exemple de charge utile suivant :

```
{
```

```
    "eventType" : "THING_GROUP_MEMBERSHIP_EVENT",
    "eventId" : "d684bd5f-6f6e-48e1-950c-766ac7f02fd1",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/
MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
}
```

Les charges utiles contiennent les attributs suivants :

eventType

Défini sur « THING_GROUP_MEMBERSHIP_EVENT ».

eventId

L'ID d'événement.

timestamp

L'horodatage UNIX du moment où l'événement s'est produit.

fonctionnement

`ADDED` lorsqu' un objet est ajouté à un groupe d'objets. `REMOVED` lorsqu'un objet est supprimé d'un groupe d'objets.

accountId

Votre Compte AWS carte d'identité.

groupArn

L'ARN du groupe d'objets.

groupId

L'ID du groupe.

thingArn

L'ARN de l'objet qui a été ajouté au groupe d'objets ou supprimé de ce groupe.

thingId

L'ID de l'objet qui a été ajouté au groupe d'objets ou supprimé de ce groupe.

membershipId

Un ID qui représente la relation entre l'objet et le groupe d'objets. Cette valeur est générée lorsque vous ajoutez un objet à un groupe d'objets.

Groupe d'objets ajouté à un groupe d'objets/retiré d'un groupe d'objets

Le registre publie les messages d'événement suivants lorsqu'un groupe d'objets est ajouté à un autre groupe d'objets ou retiré d'un autre groupe d'objets.

- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/added`
- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/removed`

Le message contient l'exemple de charge utile suivant :

```
{
    "eventType" : "THING_GROUP_HIERARCHY_EVENT",
    "eventId" : "264192c7-b573-46ef-ab7b-489fcd47da41",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "thingGroupName" : "MyRootThingGroup",
    "childGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "childGroupName" : "MyChildThingGroup"
}
```

Les charges utiles contiennent les attributs suivants :

eventType

Défini sur « THING_GROUP_HIERARCHY_EVENT ».

eventId

L'ID d'événement.

timestamp

L'horodatage UNIX du moment où l'événement s'est produit.

fonctionnement

`ADDED` lorsqu' un objet est ajouté à un groupe d'objets. `REMOVED` lorsqu'un objet est supprimé d'un groupe d'objets.

accountId

Votre Compte AWS carte d'identité.

thingGroupId

L'ID du groupe d'objets parent.

thingGroupName

Le nom du groupe d'objets parent.

childGroupId

L'ID du groupe d'objets enfant.

childGroupName

Le nom du groupe d'objets enfant.

Événements Jobs

Le service AWS IoT Jobs publie sur des rubriques réservées sur le protocole MQTT lorsque des tâches sont en attente, terminées ou annulées, et lorsqu'un appareil signale un succès ou un échec lors de l'exécution d'une tâche. Les appareils ou les applications de gestion et de surveillance peuvent suivre l'état des tâches en s'abonnant à ces rubriques.

Comment activer les événements d'emploi

Les messages de réponse du service AWS IoT Jobs ne passent pas par le courtier de messages et ne peuvent pas être souscrits par d'autres clients ou règles. Pour vous abonner aux messages liés aux activités professionnelles, utilisez les rubriques `notify` et `notify-next`. Pour plus d'informations sur les rubriques des tâches, consultez Rubriques de tâche.

Pour être informé des mises à jour des tâches, activez ces événements de tâches à l'aide de AWS Management Console, ou à l'aide de l'API ou de la CLI. Pour de plus amples informations, veuillez consulter Activez les événements pour AWS IoT.

Comment fonctionnent les événements des tâches

Comme l'annulation ou la suppression de tâches peut prendre un certain temps, deux messages sont envoyés pour indiquer le début et la fin d'une demande. Par exemple, lorsqu'une demande d'annulation démarre, un message est envoyé à la rubrique `$aws/events/job/jobID/cancellation_in_progress`. Lorsque la demande d'annulation est terminée, un message est envoyé à la rubrique `$aws/events/job/jobID/canceled`.

Le processus est le même pour une requête de suppression de tâche. Les applications de gestion et de surveillance peuvent effectuer le suivi de l'état des tâches en s'abonnant à ces rubriques. Pour plus d'informations sur la publication et l'abonnement aux rubriques MQTT, consultez the section called “Protocoles de communication des appareils”.

Types d'événements de tâche

Ce qui suit montre les différents types d'événements de tâches :

Job Completed/Canceled/Deleted

Le service AWS IoT Jobs publie un message sur un sujet MQTT lorsqu'une tâche est terminée, annulée, supprimée ou lorsque l'annulation ou la suppression sont en cours :

- `$aws/events/job/jobID/completed`
- `$aws/events/job/jobID/canceled`
- `$aws/events/job/jobID/deleted`
- `$aws/events/job/jobID/cancellation_in_progress`
- `$aws/events/job/jobID/deletion_in_progress`

Le message `completed`contient l'exemple de charge utile suivant :

```
{
  "eventType": "JOB",
  "eventId": "7364ffd1-8b65-4824-85d5-6c14686c97c6",
  "timestamp": 1234567890,
  "operation": "completed",
  "jobId": "27450507-bf6f-4012-92af-bb8a1c8c4484",
  "status": "COMPLETED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/a39f6f91-70cf-4bd2-a381-9c66df1a80d0",
```

```
    "arn:aws:iot:us-east-1:123456789012:thinggroup/2fc4c0a4-6e45-4525-
a238-0fe8d3dd21bb"
  ],
  "description": "My Job Description",
  "completedAt": 1234567890123,
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123,
  "jobProcessDetails": {
    "numberOfCanceledThings": 0,
    "numberOfRejectedThings": 0,
    "numberOfFailedThings": 0,
    "numberOfRemovedThings": 0,
    "numberOfSucceededThings": 3
  }
}
```

Le message `canceled` contient l'exemple de charge utile suivant.

```
{
  "eventType": "JOB",
  "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
  "timestamp": 1234567890,
  "operation": "canceled",
  "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
  "status": "CANCELED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
  ],
  "description": "My job description",
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123
}
```

Le message `deleted` contient l'exemple de charge utile suivant.

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
```

```
      "operation": "deleted",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETED",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Le message `cancellation_in_progress`contient l'exemple de charge utile suivant :

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "cancellation_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "CANCELLATION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Le message `deletion_in_progress`contient l'exemple de charge utile suivant :

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
```

```
      "timestamp": 1234567890,
      "operation": "deletion_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

Statut terminal de l'exécution d'une tâche

Le service AWS IoT Jobs publie un message lorsqu'un appareil met à jour l'exécution d'une tâche à l'état du terminal :

- `$aws/events/jobExecution/`*`jobID`*`/succeeded`
- `$aws/events/jobExecution/`*`jobID`*`/failed`
- `$aws/events/jobExecution/`*`jobID`*`/rejected`
- `$aws/events/jobExecution/`*`jobID`*`/canceled`
- `$aws/events/jobExecution/`*`jobID`*`/timed_out`
- `$aws/events/jobExecution/`*`jobID`*`/removed`
- `$aws/events/jobExecution/`*`jobID`*`/deleted`

Le message contient l'exemple de charge utile suivant :

```
{
  "eventType": "JOB_EXECUTION",
  "eventId": "cca89fa5-8a7f-4ced-8c20-5e653afb3572",
  "timestamp": 1234567890,
  "operation": "succeeded|failed|rejected|canceled|removed|timed_out",
  "jobId": "154b39e5-60b0-48a4-9b73-f6f8dd032d27",
  "thingArn": "arn:aws:iot:us-east-1:123456789012:myThing/6d639fbc-8f85-4a90-924d-
a2867f8366a7",
```

```
  "status": "SUCCEEDED|FAILED|REJECTED|CANCELED|REMOVED|TIMED_OUT",
  "statusDetails": {
    "key": "value"
  }
}
```

Événements du cycle de vie

AWS IoT peut publier des événements du cycle de vie sur les sujets MQTT. Ces événements sont disponibles par défaut et ne peuvent pas être désactivés.

> **Note**
>
> Les messages de cycle de vie peuvent être envoyés dans le désordre. Vous pouvez recevoir des messages en double.
> `thingName`ne sera inclus que si le client se connecte à l'aide de la fonctionnalité Exclusive Thing.

Dans cette rubrique :

- Événements de connexion/déconnexion
- Événement d'échec de tentative de connexion
- Événements d'abonnement/désabonnement

Événements de connexion/déconnexion

> **Note**
>
> Grâce à AWS IoT l'indexation du parc Device Management, vous pouvez rechercher des objets, exécuter des requêtes agrégées et créer des groupes dynamiques en fonction des Connect/Disconnect événements liés aux objets. Pour plus d'informations,veuillez consulter consultez la rubrique Fleet indexing.

AWS IoT publie un message dans les rubriques MQTT suivantes lorsqu'un client se connecte ou se déconnecte :

- `$aws/events/presence/connected/clientId` – Un client connecté à l'agent de messages.
- `$aws/events/presence/disconnected/clientId` – un client déconnecté de l'agent de messages.

Vous trouverez ci-dessous une liste des éléments JSON contenus dans les connection/disconnection messages publiés dans le `$aws/events/presence/connected/clientId` sujet.

clientId

ID du client qui se connecte ou se déconnecte.

> **Note**
>
> Les clients IDs contenant # ou + ne reçoivent aucun événement du cycle de vie.

thingName

Le nom de votre outil IoT. `thingName`ne sera inclus que si le client se connecte à l'aide de la fonctionnalité Exclusive Thing.

clientInitiatedDisconnect

True si le client est à l'origine de la déconnexion. Sinon, la valeur renvoyée est Faux. Figurant uniquement dans les messages de déconnexion.

disconnectReason

La raison pour laquelle le client se déconnecte. Figurant uniquement dans les messages de déconnexion. Le tableau suivant contient des valeurs valides et indique si le courtier enverra des messages Last Will and Testament (LWT) lorsque la déconnexion se produit.

Raison de la déconnexion	Description	Le courtier enverra les messages LWT
`AUTH_ERROR`	Le client n'a pas pu s'authentifier ou l'autorisation a échoué.	Oui. Si l'apparei l dispose d'une connexion active avant de recevoir cette erreur.

Raison de la déconnexion	Description	Le courtier enverra les messages LWT
`CLIENT_INITIATED_DISCONNECT`	Le client indique qu'il va se déconnecter. Le client peut le faire en envoyant un paquet de `DISCONNECT` contrôle MQTT ou un paquet `Close frame` s'il utilise une WebSocket connexion.	Non.
`CLIENT_ERROR`	Le client a réalisé une erreur qui a provoqué sa déconnexion. Par exemple, un client sera déconnecté pour avoir envoyé plus d'un paquet `CONNECT` MQTT sur la même connexion ou pour avoir tenté de publier avec une charge utile supérieure à la limite de cette dernière.	Oui.
`CONNECTION_LOST`	La connexion client-serveur est coupée. Cela peut se produire pendant une période de latence réseau élevée ou lorsque la connexion Internet est perdue.	Oui.
`DUPLICATE_CLIENTID`	Le client emploie un ID client déjà utilisé. Dans ce cas, le client déjà connecté sera déconnecté avec cette raison de déconnexion.	Oui.
`FORBIDDEN_ACCESS`	Le client n'est pas autorisé à être connecté. Par exemple, un client avec une adresse IP refusée échouera à se connecter.	Oui. Si l'appareil dispose d'une connexion active avant de recevoir cette erreur.
`MQTT_KEEP_ALIVE_TIMEOUT`	S'il n'y a pas de communication client-serveur pour 1.5x le temps de maintien de la connexion du client, le client est déconnecté.	Oui.
`SERVER_ERROR`	Déconnecté en raison de problèmes de serveur inattendus.	Oui.

Raison de la déconnexion	Description	Le courtier enverra les messages LWT
`SERVER_INITIATED_DISCONNECT`	Le serveur déconnecte intentionnellement un client pour des raisons opérationnelles.	Oui.
`THROTTLED`	Le client est déconnecté en raison du dépassement d'une limite de limitation.	Oui.
`WEBSOCKET_TTL_EXPIRATION`	Le client est déconnecté car WebSocket a est connecté depuis plus longtemps que sa time-to-live valeur.	Oui.
`CUSTOMAUTH_TTL_EXPIRATION`	Le client est déconnecté car il est connecté depuis plus longtemps que la time-to-live valeur de son autorisateur personnalisé.	Oui.

eventType

Type d'événement. Les valeurs valides sont `connected` ou `disconnected`.

ipAddress

Adresse IP du client de connexion. Cela peut être au IPv6 format IPv4 ou au format. Figurant uniquement dans les messages de connexion.

principalIdentifier

Informations d'identification utilisées pour l'authentification. Pour les certificats d'authentification mutuelle TLS, il s'agit de l'ID du certificat. Pour les autres connexions, ce sont les informations d'identification IAM.

sessionIdentifier

Un identifiant unique au monde AWS IoT qui existe pendant toute la durée de la session.

timestamp

Une approximation du moment où l'événement s'est produit.

versionNumber

Numéro de version de l'événement de cycle de vie. Il s'agit d'un entier qui augmente de façon monotone pour chaque connexion d'ID client. Le numéro de version peut être utilisé par un abonné afin de déduire l'ordre des événements de cycle de vie.

> **Note**
>
> Les messages de connexion et de déconnexion d'une connexion de client ont le même numéro de version.
> Le numéro de version peut ignorer des valeurs ; il n'est pas garanti qu'il augmentera uniformément de 1 à chaque événement.
> Si un client n'est pas connecté pendant une heure environ, le numéro de version est réinitialisé à 0. Pour les sessions persistantes, le numéro de version est remis à 0 après qu'un client a été déconnecté plus longtemps que le délai configuré time-to-live (TTL) pour la session persistante.

Un message de connexion présente la structure suivante.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1573002230757,
    "eventType": "connected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "ipAddress": "192.0.2.0",
    "versionNumber": 0
}
```

Un message de déconnexion présente la structure suivante.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1573002340451,
    "eventType": "disconnected",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
```

```
    "clientInitiatedDisconnect": true,
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
    "versionNumber": 0
}
```

Gestion des déconnexions du client

Les bonnes pratiques consistent à toujours implémenter un état d'attente pour les événements du cycle de vie, notamment avec des messages LWT (Last Will and Testament). À réception d'un message de déconnexion, votre code doit déclencher un délai d'attente et vérifier si un appareil est toujours hors ligne avant de prendre des mesures. Pour ce faire, vous pouvez utiliser des files d'attente à retardement SQS. Lorsqu'un client reçoit un message LWT ou un événement de cycle de vie, vous pouvez mettre ce message en file d'attente (pendant 5 secondes, par exemple). Lorsque ce message est disponible et traité (par Lambda ou un autre service), vous pouvez commencer par vérifier si l'appareil est toujours hors connexion avant de prendre d'autres mesures.

Événement d'échec de tentative de connexion

AWS IoT publie un message dans la rubrique MQTT suivante lorsqu'un client n'est pas autorisé à se connecter ou lorsqu'un dernier testament est configuré et que le client n'est pas autorisé à publier sur cette rubrique testamentaire.

```
$aws/events/presence/connect_failed/clientId
```

Vous trouverez ci-dessous une liste des éléments JSON contenus dans les messages d'autorisation de connexion publiés dans le `$aws/events/presence/connect_failed/clientId` sujet.

clientId

ID client du client qui a tenté de se connecter sans succès.

> **Note**
>
> Les clients IDs contenant # ou + ne reçoivent aucun événement du cycle de vie.

thingName

Le nom de votre outil IoT. `thingName`ne sera inclus que si le client se connecte à l'aide de la fonctionnalité Exclusive Thing.

timestamp

Une approximation du moment où l'événement s'est produit.

eventType

Type d'événement. La valeur valide est`connect_failed`.

connectFailureReason

La raison pour laquelle la connexion échoue. La valeur valide est`AUTHORIZATION_FAILED`.

principalIdentifier

Informations d'identification utilisées pour l'authentification. Pour les certificats d'authentification mutuelle TLS, il s'agit de l'ID du certificat. Pour les autres connexions, ce sont les informations d'identification IAM.

sessionIdentifier

Un identifiant unique au monde AWS IoT qui existe pendant toute la durée de la session.

ipAddress

Adresse IP du client de connexion. Cela peut être au IPv6 format IPv4 ou au format. Figurant uniquement dans les messages de connexion.

La structure d'un message d'échec de connexion est la suivante.

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1460065214626,
    "eventType": "connect_failed",
    "connectFailureReason": "AUTHORIZATION_FAILED",
    "principalIdentifier": "12345678901234567890123456789012",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "ipAddress" : "192.0.2.0"
}
```

Événements d'abonnement/désabonnement

AWS IoT publie un message dans la rubrique MQTT suivante lorsqu'un client s'abonne ou se désabonne à une rubrique MQTT :

```
$aws/events/subscriptions/subscribed/clientId
```

or

```
$aws/events/subscriptions/unsubscribed/clientId
```

Où `clientId` est l'ID du client MQTT qui se connecte à l'agent de messages AWS IoT .

Le message publié sur cette rubrique a la structure suivante :

```
{
    "clientId": "186b5",
    "thingName": "exampleThing",
    "timestamp": 1460065214626,
    "eventType": "subscribed" | "unsubscribed",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "12345678901234567890123456789012",
    "topics" : ["foo/bar","device/data","dog/cat"]
}
```

Voici une liste d'éléments JSON qui sont contenus dans les messages d'abonnement/désabonnement publiés dans les rubriques `$aws/events/subscriptions/subscribed/clientId` et `$aws/events/subscriptions/unsubscribed/clientId`.

clientId

ID du client qui s'abonne ou se désabonne.

> **Note**
>
> Les clients IDs contenant # ou + ne reçoivent aucun événement du cycle de vie.

thingName

Le nom de votre outil IoT. `thingName`ne sera inclus que si le client se connecte à l'aide de la fonctionnalité Exclusive Thing.

eventType

Type d'événement. Les valeurs valides sont `subscribed` ou `unsubscribed`.

principalIdentifier

Informations d'identification utilisées pour l'authentification. Pour les certificats d'authentification mutuelle TLS, il s'agit de l'ID du certificat. Pour les autres connexions, ce sont les informations d'identification IAM.

sessionIdentifier

Un identifiant unique au monde AWS IoT qui existe pendant toute la durée de la session.

timestamp

Une approximation du moment où l'événement s'est produit.

topics

Tableau des sujets MQTT auquel le client s'est abonné.

> **Note**
>
> Les messages de cycle de vie peuvent être envoyés dans le désordre. Vous pouvez recevoir des messages en double.

Résolution des problèmes AWS IoT

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

Les informations suivantes peuvent vous aider à résoudre les problèmes courants dans AWS IoT.

Tâches

- AWS IoT Core guide de dépannage
- AWS IoT Device Management guide de dépannage
- AWS IoT Guide de dépannage de Device Advisor
- AWS IoT erreurs

AWS IoT Core guide de dépannage

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

Il s'agit de la section de résolution des problèmes pour AWS IoT Core.

Rubriques

- Diagnostic des problèmes de connectivité
- Diagnostic des problèmes de règles
- Diagnostic des problèmes de shadows
- Diagnostic des problèmes liés aux actions de flux d'entrée Salesforce IoT
- Diagnostic des limites de débit
- Résolution des problèmes de déconnexion du parc d'appareils

Diagnostic des problèmes de connectivité

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

Une connexion réussie à AWS IoT nécessite :

- Une connexion valide
- Un certificat valide et actif
- Une politique qui autorise la connexion et le fonctionnement souhaités

Connexion

Comment trouver le bon point de terminaison ?

- L'adresse `endpointAddress` renvoyée par `aws iot describe-endpoint --endpoint-type iot:Data-ATS`

  or

- L'adresse `domainName` renvoyée par `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"`

Comment puis-je trouver la valeur SNI (Server Name Indication) correcte ?

La valeur SNI correcte est celle `endpointAddress` renvoyée par describe-endpoint ou `domainName` renvoyée par les describe-domain-configuration commandes. Il s'agit de la même adresse que celle du point de terminaison de l'étape précédente. Lorsqu'ils connectent des appareils à AWS IoT Core, les clients peuvent envoyer l'extension SNI (Server Name Indication), ce qui n'est pas obligatoire mais fortement recommandé. Pour utiliser des fonctionnalités telles que l'enregistrement multi-comptes, les domaines personnalisés et les points de terminaison d'un VPC, vous devez utiliser l'extension SNI. Pour plus d'informations, voir Sécurité du transport dans AWS IoT.

Comment résoudre un problème de connectivité qui persiste ?

Vous pouvez utiliser AWS Device Advisor pour résoudre les problèmes. Les tests prédéfinis de Device Advisor vous aident à valider le logiciel de votre appareil par rapport aux meilleures

pratiques en matière d'utilisation des protocoles TLS, MQTT, AWS IoT Device Shadow, et AWS IoT IoT Jobs.

Voici un lien vers le contenu existant de Device Advisor.

Authentification

Les appareils doivent être authentifiés pour se connecter aux points de AWS IoT terminaison. Pour les appareils utilisés Certificats client X.509 pour l'authentification, les certificats doivent être enregistrés AWS IoT et actifs.

Comment mes appareils authentifient-ils les AWS IoT terminaux ?

Ajoutez le certificat AWS IoT CA au trust store de votre client. Reportez-vous à la documentation sur l'authentification de serveur dans AWS IoT Core et suivez les liens pour télécharger le certificat CA approprié.

Qu'est-ce qui est vérifié lorsqu'un appareil se connecte à AWS IoT ?

Lorsqu'un appareil tente de se connecter à AWS IoT :

1. AWS IoT vérifie la validité du certificat et de la valeur de l'indication du nom du serveur (SNI).
2. AWS IoT vérifie que le certificat utilisé est enregistré auprès du AWS IoT compte et qu'il a été activé.
3. Lorsqu'un appareil tente d'effectuer une action AWS IoT, telle que s'abonner ou publier un message, la politique attachée au certificat utilisé pour se connecter est vérifiée pour confirmer que le terminal est autorisé à effectuer cette action.

Comment puis-je valider un certificat correctement configuré ?

Utilisez la commande OpenSSL `s_client` pour tester une connexion à un point de terminaison AWS IoT :

```
openssl s_client -connect custom_endpoint.iot.aws-region.amazonaws.com:8443 -
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

Pour plus d'informations sur l'utilisation d'`openssl s_client`, consultez la documentation OpenSSL s_client.

Comment puis-je vérifier le statut d'un certificat ?

- Répertoriez les certificats.

Si vous ne connaissez pas l'ID de certificat, vous pouvez consulter l'état de tous vos certificats à l'aide de la `aws iot list-certificates` commande.

- Afficher les détails d'un certificat

  Si vous connaissez l'ID du certificat, cette commande affiche des informations plus détaillées sur le certificat.

  ```
  aws iot describe-certificate --certificate-id "certificateId"
  ```

- Vérifiez le certificat dans la AWS IoT console

  Dans la AWS IoT console, dans le menu de gauche, choisissez Sécurisé, puis Certificats.

  Choisissez le certificat dans la liste pour ouvrir sa page de détail..

  Sur la page détaillée du certificat, vous pouvez voir son statut actuel.

  Le statut du certificat peut être modifié à l'aide du menu Actions en haut à droite de la page de détails.

Autorisation

AWS IoT ressources utilisées AWS IoT Core politiques pour autoriser ces ressources à effectuer des actions. Pour qu'une action soit autorisée, les AWS IoT ressources spécifiées doivent être associées à un document de politique autorisant l'exécution de cette action.

J'ai reçu une réponse PUBNACK ou SUBNACK de l'agent. Que puis-je faire ?

Assurez-vous qu'une politique est attachée au certificat que vous utilisez pour appeler AWS IoT. Toutes les opérations de publication/abonnement sont rejetées par défaut.

Assurez-vous que la politique ci-jointe autorise les actions que vous essayez d'effectuer.

Assurez-vous que la politique ci-jointe autorise les ressources qui tentent d'effectuer les actions autorisées.

J'ai une entrée AUTHORIZATION_FAILURE dans mes journaux.

Assurez-vous qu'une politique est attachée au certificat que vous utilisez pour appeler AWS IoT. Toutes les opérations de publication/abonnement sont rejetées par défaut.

Assurez-vous que la politique ci-jointe autorise les actions que vous essayez d'effectuer.

Assurez-vous que la politique ci-jointe autorise les ressources qui tentent d'effectuer les actions autorisées.

Comment puis-je vérifier ce que la politique autorise ?

Dans le menu de gauche de la AWS IoT console, choisissez Sécurité, puis Certificats.

Choisissez le certificat dans la liste pour ouvrir sa page de détail..

Sur la page détaillée du certificat, vous pouvez voir son statut actuel.

Dans le menu de gauche de la page de détails du certificat, choisissez Politiques pour voir les politiques attachées au certificat.

Choisissez la politique de votre choix afin d'afficher sa page de détails.

Sur la page de détails de la politique, consultez le document de politique de la politique pour voir ce qu'il autorise.

Choisissez Modifier le document de stratégie pour apporter des modifications au document de stratégie.

Sécurité et identité

Lorsque vous fournissez les certificats de serveur pour une configuration de domaine AWS IoT personnalisée, les certificats ont un maximum de quatre noms de domaine.

Pour plus d’informations, consultez Points de terminaison et quotas AWS IoT Core.

Diagnostic des problèmes de règles

> Aidez-nous à améliorer ce sujet
>
> Dites-nous ce qui pourrait contribuer à l'améliorer

Cette section décrit certaines des choses à vérifier lorsque vous rencontrez un problème avec une règle.

Configuration des CloudWatch journaux pour le dépannage

La meilleure façon de résoudre les problèmes que vous rencontrez avec les règles est d'utiliser les CloudWatch journaux. Lorsque vous activez CloudWatch Logs for AWS IoT, vous pouvez voir quelles règles sont déclenchées, ainsi que leur succès ou leur échec. Vous obtenez également des informations concernant la correspondance ou non des conditions de clause WHERE. Pour de plus amples informations, veuillez consulter Surveiller AWS IoT à l'aide CloudWatch des journaux.

Le problème le plus fréquent avec les règles est celui de l'autorisation. Les journaux indiquent si votre rôle n'est pas autorisé à jouer AssumeRole sur la ressource. Voici un exemple de journal généré par la journalisation affinée :

```
{
    "timestamp": "2017-12-09 22:49:17.954",
    "logLevel": "ERROR",
    "traceId": "ff563525-6469-506a-e141-78d40375fc4e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecution",
    "clientId": "iotconsole-123456789012-3",
    "topicName": "test-topic",
    "ruleName": "rule1",
    "ruleAction": "DynamoAction",
    "resources": {
        "ItemHashKeyField": "id",
        "Table": "trashbin",
        "Operation": "Insert",
        "ItemHashKeyValue": "id",
        "IsPayloadJSON": "true"
    },
    "principalId": "ABCDEFG1234567ABCD890:outis",
    "details": "User: arn:aws:sts::123456789012:assumed-role/dynamo-
testbin/5aUMInJH is not authorized to perform: dynamodb:PutItem on
 resource: arn:aws:dynamodb:us-east-1:123456789012:table/testbin (Service:
 AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException; Request ID:
 AKQJ987654321AKQJ123456789AKQJ987654321AKQJ987654321)"
}
```

Voici un exemple similaire de journal généré par la journalisation globale :

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFG1234567ABCD890:outis [ERROR] EVENT:DynamoActionFailure
```

```
TOPICNAME:test-topic CLIENTID:iotconsole-123456789012-3
MESSAGE:Dynamo Insert record failed. The error received was User:
 arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJI is not authorized to
 perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-east-1:123456789012:table/
testbin
(Service: AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException;
 Request ID: AKQJ987654321AKQJ987654321AKQJ987654321AKQJ987654321).
Message arrived on: test-topic, Action: dynamo, Table: trashbin, HashKeyField: id,
 HashKeyValue: id, RangeKeyField: None, RangeKeyValue: 123456789012
No newer events found at the moment. Retry.
```

Pour de plus amples informations, veuillez consulter the section called “Afficher AWS IoT les journaux dans la CloudWatch console”.

Diagnostic des services externes

Les services externes sont contrôlés par l'utilisateur final. Avant d'exécuter une règle, assurez-vous que les services externes que vous avez liés à votre règle sont configurés et disposent de suffisamment d'unités de débit et de capacité pour votre application.

Diagnostic des problèmes SQL

Si votre requête SQL ne renvoie pas les données attendues :

- Consultez les journaux pour détecter les messages d'erreur.
- Vérifiez que votre syntaxe SQL correspond au document JSON contenu dans le message.

  Passez en revue les noms d'objets et de propriétés utilisés dans la requête avec ceux utilisés dans le document JSON de la charge utile des messages du sujet. Pour plus d'informations sur le formatage JSON dans les requêtes SQL, voir Extensions JSON.
- Vérifiez si les noms d'objets ou de propriétés JSON incluent des caractères réservés ou numériques.

  Pour plus d'informations sur les caractères réservés dans les références d'objets JSON dans les requêtes SQL, consultez Extensions JSON.

Diagnostic des problèmes de shadows

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

Diagnostic de shadows

Problème	Consignes pour la résolution des problèmes
Un document shadow d'appareil est rejeté avec `Invalid JSON document`.	Si vous ne connaissez pas JSON, modifiez les exemples fournis dans ce manuel pour les adapter à votre utilisation. Pour de plus amples informations, veuillez consulter Exemples de documents shadow.
J'ai envoyé un code JSON correct, mais aucune ou seulement quelques parties sont stockées dans le document shadow d'appareil.	Vérifiez que vous avez suivi les consignes de formatage JSON. Seuls les champs JSON des sections `desired` et `reported` sont stockés. Le contenu JSON à l'extérieur de ces sections est ignoré (même s'il est formaté correctement).
J'ai reçu un message d'erreur indiquant que le shadow d'appareil dépasse la taille autorisée.	Le shadow d'appareil prend en charge seulement 8 Ko de données. Essayez de raccourcir les noms de champs au sein de votre document JSON ou créez simplement des shadows supplémentaires en créant plus d'objets. Un appareil peut avoir un nombre illimité d'objets/de shadows associés. La seule condition est que chaque nom d'objet soit unique dans votre compte.
Lorsque je reçois un shadow d'appareil, celui-ci fait plus de 8 Ko. Comment est-ce possible ?	Dès réception, le AWS IoT service ajoute des métadonnées à l'ombre de l'appareil. Le service inclut ces données dans sa réponse, mais elles ne comptent pas dans la limite de 8 Ko. Seules les données d'état `desired` et `reported` au

Problème	Consignes pour la résolution des problèmes
	sein du document d'état envoyé au shadow d'appareil comptent pour le calcule de la limite.
Ma demande a été rejetée car la version était incorrecte. Que dois-je faire ?	Exécutez une opération GET pour effectuer une synchronisation avec la dernière version du document d'état. Lors de l'utilisation de MQTT, abonnez-vous à la rubrique /update/accepted pour recevoir des notifications concernant les changements d'état et recevoir la dernière version du document JSON.
L'horodatage est décalé de quelques secondes.	L'horodatage des champs individuels et de l'ensemble du document JSON est mis à jour lorsque le document est reçu par le AWS IoT service ou lorsque le document d'état est publié sur le. /update/accepted and ./update/deltamessage. Les messages peuvent être retardés sur le réseau, ce qui peut entraîner un décalage de l'horodatage de quelques secondes.
Mon appareil peut publier et s'abonner aux rubriques de shadow correspondantes, mais lorsque je tente de mettre à jour le document de shadow via l'API HTTP REST, un message HTTP 403 s'affiche.	Assurez-vous d'avoir créé des politiques dans IAM pour autoriser l'accès à ces rubriques et à l'action correspondante (UPDATE/GET/DELETE) pour les informations d'identification que vous utilisez. Les politiques IAM et les politiques de certification sont indépendantes.
Autres problèmes.	Le service Device Shadow enregistre les erreurs dans CloudWatch Logs. Pour identifier les problèmes liés au périphérique et à la configuration, activez CloudWatch les journaux et consultez les journaux pour obtenir des informations de débogage.

Diagnostic des problèmes liés aux actions de flux d'entrée Salesforce IoT

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

Trace d'exécution

Comment consulter la trace d'exécution d'une action Salesforce ?

Consultez la section Surveiller AWS IoT à l'aide CloudWatch des journaux. Après avoir activé les journaux, vous pouvez consulter la trace d'exécution de l'action Salesforce.

Succès et échec d'une action

Comment vérifier que des messages ont été correctement envoyés à un flux d'entrée Salesforce IoT ?

Consultez les journaux générés par l'exécution de l'action Salesforce dans CloudWatch Logs. Si vous voyez cela`Action executed successfully`, cela signifie que le moteur de AWS IoT règles a reçu la confirmation de Salesforce IoT indiquant que le message a été correctement transmis au flux d'entrée ciblé.

Si vous rencontrez des problèmes avec la plateforme Salesforce IoT, consultez le support Salesforce IoT.

Que faire si des messages ne sont pas correctement envoyés à un flux d'entrée Salesforce IoT ?

Consultez les journaux générés par l'exécution de l'action Salesforce dans CloudWatch Logs. Selon la nature de l'entrée du journal, vous pouvez tenter les opérations suivantes :

`Failed to locate the host`

Vérifiez que le paramètre `url` de l'action est correct et que le flux d'entrée Salesforce IoT Input existe bien.

`Received Internal Server Error from Salesforce`

Réessayer. Si le problème persiste, contactez le support Salesforce IoT.

`Received Bad Request Exception from Salesforce`

Vérifiez qu'il n'y a pas d'erreurs dans la charge utile que vous envoyez.

`Received Unsupported Media Type Exception from Salesforce`

Salesforce IoT ne prend pas en charge les charges utiles binaires pour le moment. Vérifiez que vous envoyez bien une charge utile JSON.

`Received Unauthorized Exception from Salesforce`

Vérifiez que le paramètre `token` de l'action est correct et que votre jeton est toujours valide.

`Received Not Found Exception from Salesforce`

Vérifiez que le paramètre `url` de l'action est correct et que le flux d'entrée Salesforce IoT Input existe bien.

Si vous recevez un message d'erreur qui n'est pas répertorié ici, contactez le AWS IoT Support.

Diagnostic des limites de débit

Dépannage "Limite de flux dépassée pour votre AWS compte"

Si vous voyez `"Error: You have exceeded the limit for the number of streams in your AWS account."`, vous pouvez nettoyer les flux inutilisés de votre compte au lieu de demander une augmentation de la limite.

Pour nettoyer un flux inutilisé que vous avez créé à l'aide du SDK AWS CLI ou :

```
aws iot delete-stream –stream-id value
```

Pour plus de détails, consultez delete-stream.

> **Note**
>
> Vous pouvez utiliser la `list-streams` commande pour trouver le flux IDs.

Résolution des problèmes de déconnexion du parc d'appareils

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

AWS IoT les déconnexions du parc d'appareils peuvent se produire pour plusieurs raisons. Cet article explique comment diagnostiquer une raison de déconnexion et comment gérer les déconnexions causées par la maintenance régulière du AWS IoT service ou une limite de limitation.

Pour diagnostiquer la raison de la déconnexion

Vous pouvez vérifier le groupe de journaux AWSIotLogSv2 CloudWatchpour identifier la raison de la déconnexion dans le `disconnectReason` champ de l'entrée du journal.

Vous pouvez également utiliser AWS IoT la fonction d'événements du cycle de vie pour identifier la raison de la déconnexion. Si vous êtes abonné à l'événement de déconnexion de Lifecycle (`$aws/events/presence/disconnected/`*`clientId`*), vous recevrez une notification AWS IoT lorsque la déconnexion aura lieu. Vous pouvez identifier la raison de la déconnexion dans le `disconnectReason` champ de la notification.

Pour plus d'informations, consultez les rubriques Entrées du CloudWatch AWS IoT journal et Événements du cycle de vie.

Pour résoudre les problèmes de déconnexion dus à AWS IoT la maintenance du service

Les déconnexions causées par AWS IoT la maintenance du service sont enregistrées comme `SERVER_INITIATED_DISCONNECT` un événement AWS IoT du cycle de vie et CloudWatch. Pour gérer ces déconnexions, ajustez votre configuration côté client pour vous assurer que vos appareils peuvent être automatiquement reconnectés à la plateforme. AWS IoT

Pour résoudre les problèmes de déconnexion dus à une limite d'étranglement

Les déconnexions causées par une limite de limitation sont enregistrées `THROTTLED` en tant qu'événement AWS IoT du cycle de vie et. CloudWatch Pour gérer ces déconnexions, vous pouvez demander que la limite du courtier de messages augmente à mesure que le nombre d'appareils augmente.

Pour de plus amples informations, veuillez consulter AWS IoT Core Message Broker.

AWS IoT Device Management guide de dépannage

> Aidez-nous à améliorer ce sujet
>
> Dites-nous ce qui pourrait contribuer à l'améliorer

Il s'agit de la section de résolution des problèmes pour AWS IoT Device Management.

Rubriques

- AWS IoT Dépannage des tâches
- Résolution des problèmes liés à l'indexation du parc
- AWS IoT Résolution des problèmes liés au catalogue des packages logiciels de gestion des appareils

AWS IoT Dépannage des tâches

Il s'agit de la section de résolution des problèmes pour AWS IoT Jobs.

Comment localiser un point de terminaison AWS IoT Jobs ?

Comment localiser le point de terminaison du plan de contrôle des AWS IoT tâches ?

AWS IoT Jobs prend en charge les opérations de l'API du plan de contrôle à l'aide du protocole HTTPS. Vérifiez que vous vous êtes connecté au point de terminaison du plan de contrôle approprié à l'aide du protocole HTTPS.

Pour obtenir la liste des points de terminaison AWS spécifiques à une région, voir Points de terminaison du plan de contrôle AWS IoT du noyau.

Pour obtenir la liste des points de terminaison du AWS IoT plan de contrôle des tâches conformes à la norme FIPS, voir Points de terminaison FIPS par service

> Note
>
> AWS IoT Emplois et AWS IoT Core partage les mêmes points de terminaison AWS spécifiques à une région.

Comment localiser le point de terminaison du plan de données AWS IoT Jobs ?

AWS IoT Jobs prend en charge les opérations d'API du plan de données à l'aide des protocoles HTTPS et MQTT. Vérifiez que vous vous êtes connecté au point de terminaison du plan de données approprié à l'aide du protocole HTTPS ou MQTT.

- Protocole : HTTP
  - Utilisez la commande describe-endpointCLI ci-dessous ou l'`DescribeEndpoint`API REST. Pour le type de point de terminaison, utilisez `iot:Jobs`.

    ```
    aws iot describe-endpoint --endpoint-type iot:Jobs
    ```
- Protocole MQTT
  - Utilisez la commande describe-endpointCLI ci-dessous ou l'`DescribeEndpoint`API REST. Pour le type de point de terminaison, utilisez `iot:Data-ATS`.

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

Pour une liste des points de terminaison du AWS IoT plan de données des Tâches conformes à la norme FIPS, voir Points de terminaison FIPS par service

Comment puis-je surveiller l'activité AWS IoT des offres d'emploi et fournir des statistiques ?

Le suivi de l'activité des AWS IoT Jobs à l'aide d'Amazon CloudWatch fournit une visibilité en temps réel des opérations AWS IoT Jobs en cours et permet de contrôler les coûts grâce à des CloudWatch alarmes via AWS IoT des règles. Vous devez configurer la journalisation avant de pouvoir surveiller l'activité AWS IoT des tâches et configurer des CloudWatch alarmes. Pour plus d'informations sur la mise en place de la journalisation, consultez Configuration de la AWS IoT journalisation.

Pour plus d'informations sur Amazon CloudWatch et sur la manière de configurer l'autorisation via un rôle d'utilisateur IAM pour utiliser les CloudWatch ressources, consultez Gestion des identités et des accès pour Amazon CloudWatch.

Comment configurer les statistiques et le suivi AWS IoT des offres d'emploi à l'aide d'Amazon CloudWatch ?

Pour configurer la AWS IoT journalisation, suivez les étapes décrites dans Configurer la AWS IoT journalisation. AWS IoT la configuration de la journalisation peut être effectuée dans l'API

AWS Management Console AWS CLI, ou. AWS IoT la configuration de la journalisation pour des groupes d'objets spécifiques doit être effectuée uniquement dans l'API AWS CLI or.

La section Mesures relatives aux AWS IoT tâches contient les mesures relatives aux AWS IoT tâches utilisées pour surveiller l'activité AWS IoT des tâches. Il explique comment afficher les métriques dans le AWS Management Console et AWS CLI.

En outre, vous pouvez configurer des CloudWatch alarmes pour vous avertir des mesures spécifiques que vous souhaitez surveiller de près. Pour obtenir des conseils sur la configuration des alarmes, consultez la section Utilisation des CloudWatch alarmes Amazon.

Gestion des flottes d'appareils et résolution des problèmes liés à un seul appareil

L'exécution d'une tâche conserve un statut **QUEUED** indéfini

Lorsqu'une exécution de tâche dont l'état d'état est égal à `QUEUED` ne passe pas à l'état logique suivant, tel que `IN_PROGRESS`, `FAILED`, ou `TIMED_OUT`, l'un des scénarios suivants peut en être la cause :

- Passez en revue l'activité de votre appareil dans les CloudWatch journaux situés dans la CloudWatch console. Pour plus d'informations, reportez-vous à la section Surveillance à AWS IoT l'aide CloudWatch des journaux.
- Le rôle IAM associé à la tâche et à son exécution ultérieure peut ne pas disposer des autorisations correctes répertoriées dans l'une des déclarations de politique de la stratégie IAM attachée à ce rôle IAM. Utilisez l'`describe-job`API pour identifier le rôle IAM lié à cette tâche et à son exécution ultérieure, et vérifiez la politique IAM pour connaître les autorisations correctes. Une fois que les déclarations d'autorisation de politique ont été mises à jour, vous devriez être en mesure d'exécuter la commande `AssumeRole`API sur la ressource.

Aucune exécution de tâche n'a été créée pour mon objet ou mon groupe d'objets

Lorsqu'une tâche met à jour son statut`IN_PROGRESS`, elle commence à déployer le document de tâche sur tous les appareils de votre groupe cible. Cette mise à jour de l'état créera une exécution de tâche pour chaque équipement cible. Si aucune exécution de tâche n'a été créée pour l'un des équipements cibles, reportez-vous aux instructions suivantes :

- La tâche est-elle `thing` directement ciblée par la tâche, la tâche a-t-elle un statut égal `IN_PROGRESS` à et la tâche est-elle simultanée ? Si les trois conditions sont remplies, la tâche

continue d'envoyer des exécutions de tâches à tous les appareils de votre groupe cible et cette tâche spécifique n'`thing`a pas encore été exécutée.

- Vérifiez les appareils de votre groupe cible pour connaître la tâche et l'état de l'état de la tâche dans la console de AWS gestion ou utilisez la commande `describe-job`API.
- Utilisez la commande `describe-job`API pour vérifier si la `IsConcurrent` propriété de la tâche est définie sur true ou false. Pour plus d'informations, consultez Job Limits.

- Le n'`thing` est pas directement visé par la tâche.
  - Si le `Thing` a été ajouté à un `ThingGroup` et que la tâche le ciblait`ThingGroup`, vérifiez `Thing` qu'il fait partie du `ThingGroup`.
  - S'il s'agit d'un travail de `IN_PROGRESS` capture instantanée dont le statut est simultané, le travail envoie toujours des exécutions de tâches à tous les appareils de votre groupe cible et cette tâche spécifique n'`Thing`a pas encore été exécutée.
  - S'il s'agit d'une tâche continue dont le statut de `IN_PROGRESS` et est simultané, la tâche continue d'envoyer des exécutions de tâches à tous les appareils de votre groupe cible et cette tâche spécifique n'`Thing`a pas encore été exécutée. Pour les tâches continues uniquement, vous pouvez également supprimer le `Thing` du, `ThingGroup` puis le `Thing` rajouter au`ThingGroup`.
  - S'il s'agit d'une tâche instantanée dont l'état de `IN_PROGRESS` statut est non simultané, il est probable que la relation `ThingGroup` d'adhésion `Thing` ou d'adhésion ne soit pas reconnue par AWS IoT Jobs. Il est recommandé d'ajouter quelques secondes d'attente après votre `AddThingToThingGroup` appel avant de créer votre`Job`. Vous pouvez également passer à la sélection cible`Continuous`, ce qui permettra au service de remplacer l'événement retardé `Thing` et de rattachement de l'`ThingGroup`adhésion.

La nouvelle tâche échoue en raison d'une **`LimitedExceededException`**erreur

Si la création de votre tâche échoue avec une réponse d'erreur de`LimitedExceededException`, appelez l'`list-jobs`API et passez en revue toutes les tâches `isConcurrent=true` pour déterminer si vous avez atteint votre limite de simultanéité des tâches. Voir Limites de tâches pour plus d'informations sur les tâches simultanées. Pour consulter vos limites de simultanéité de tâches et pour demander une augmentation de la limite, consultez AWS IoT Device Management Limites et quotas de tâches.

Limite de taille du document

La taille du document de travail est limitée par la taille de la charge utile MQTT. Si vous avez besoin d'un document de travail supérieur à 32 kB (kilo-octets) ou 32 000 Go (octets), créez et stockez le document de travail dans Amazon S3 et ajoutez l'URL d'un objet Amazon S3 dans le `documentSource` champ de l'`CreateJob`API ou à l'aide du AWS CLI. Pour le AWS Management Console, ajoutez l'URL d'un objet Amazon S3 dans la zone de texte URL Amazon S3 lors de la création d'une tâche.

- AWS Management Console créer de la documentation sur les tâches : créez et gérez les tâches à l'aide du AWS Management Console
- AWS CLI créer de la documentation sur les tâches : créez et gérez les tâches à l'aide du AWS CLI
- `CreateJob`Documentation de l'API : CreateJob

Les messages MQTT côté appareil demandent des limites de limitation

Si vous recevez un code d'erreur 400 `ThrottlingException`, le message MQTT côté appareil a échoué car la limite de demandes simultanées côté appareil a été atteinte. Consultez les AWS IoT Device Management limites et quotas des tâches pour plus d'informations sur les limites d'accélération et pour savoir si elles sont ajustables.

Erreur de délai de connexion

Un code d'erreur 400 `RequestExpired` indique un échec de connexion dû à une latence élevée ou à de faibles valeurs de délai d'attente côté client.

- Voir Tester la connectivité avec le point de terminaison de données de votre appareil pour plus d'informations sur le test de la connexion entre le côté client et le côté serveur.

Commande d'API non valide

Vérifiez que la bonne commande d'API a été saisie pour éviter un message d'erreur indiquant que la commande d'API n'est pas valide. Consultez la AWS IoT Référence d'API pour une liste complète de toutes les commandes AWS IoT d'API.

Erreur de connexion côté service

Un code d'erreur 503 `ServiceUnavailable` indique que l'erreur provient du côté serveur.

- Voir AWS Health Dashboard (tous les AWS services) pour connaître l'état actuel de tous les AWS services.
- Voir AWS Health Dashboard (personnel Compte AWS) pour le statut actuel de votre compte personnel Compte AWS.

Résolution des problèmes liés à l'indexation du parc

Dépannage des requêtes d'agrégation pour le service d'indexation de parc

Si vous rencontrez des erreurs de non-concordance de type, vous pouvez utiliser CloudWatch les journaux pour résoudre le problème. CloudWatch Les journaux doivent être activés avant que les journaux ne soient écrits par le service Fleet Indexing. Pour de plus amples informations, veuillez consulter Surveiller AWS IoT à l'aide CloudWatch des journaux.

Lorsque vous effectuez des requêtes d'agrégation sur des champs non gérés, vous pouvez uniquement spécifier un champ que vous avez défini dans `customFields` l'argument passé à `UpdateIndexingConfiguration` ou update-indexing-configuration. Si la valeur du champ n'est pas cohérente avec le type de données du champ configuré, cette valeur est ignorée lorsque vous effectuez une requête d'agrégation.

Si un champ ne peut pas être indexé en raison d'un type non concordant, le service Fleet Indexing envoie un journal des erreurs à Logs. CloudWatch Le journal des erreurs contient le nom du champ, la valeur qui n'a pas pu être convertie et le nom d'objet de l'appareil. Voici un exemple de journal des erreurs.

```
{
  "timestamp": "2017-02-20 20:31:22.932",
  "logLevel": "ERROR",
  "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
  "accountId": "000000000000",
  "status": "SucceededWithIssues",
  "eventType": "IndexingCustomFieldFailed",
  "thingName": "thing0",
  "failedCustomFields": [
    {
      "Name": "attributeName1",
      "Value": "apple",
      "ExpectedType": "String"
    },
```

```
    {
      "Name": "attributeName2",
      "Value": "2",
      "ExpectedType": "Boolean"
    }
  ]
}
```

Si un appareil a été déconnecté pendant environ une heure, la valeur `timestamp` du statut de connectivité peut être manquante. Pour les sessions persistantes, la valeur peut être manquante lorsqu'un client a été déconnecté plus longtemps que le délai configuré time-to-live (TTL) pour la session persistante. Les données de statut de connectivité sont indexées uniquement pour les connexions où l'ID client contient un nom d'objet correspondant. (L'ID client est la valeur utilisée pour connecter un appareil AWS IoT Core.)

Résolution de problèmes de configuration d'indexation de la flotte

Impossible de rétrograder la configuration d'indexation du parc

La rétrogradation de la configuration d'indexation de flotte n'est pas prise en charge lorsque vous souhaitez supprimer les sources de données associées à une métrique de flotte ou à un groupe dynamique.

Par exemple, si votre configuration d'indexation contient des données de registre, des données fictives et des données de connectivité, et qu'une métrique de flotte existe avec la requête`thingName:TempSensor* AND shadow.desired.temperature>80`, la mise à jour de la configuration d'indexation pour inclure uniquement les données du registre entraînera une erreur.

La modification des champs personnalisés utilisés par les indicateurs de flotte existants n'est pas prise en charge.

Impossible de mettre à jour votre configuration d'indexation en raison de mesures de flotte incompatibles ou de groupes dynamiques

Si vous ne pouvez pas mettre à jour votre configuration d'indexation en raison de mesures de flotte ou de groupes dynamiques incompatibles, supprimez les mesures de flotte ou les groupes dynamiques incompatibles avant de mettre à jour la configuration d'indexation.

Résolution des problèmes liés à l'indexation des emplacements et aux géorequêtes

Pour résoudre les erreurs de type non concordantes dans l'indexation des emplacements et les géorequêtes, vous pouvez activer les journaux. CloudWatch Pour plus d'informations sur le suivi de AWS IoT l'utilisation CloudWatch, suivez le step-by-step guide.

Lorsque vous indexez des données de localisation à l'aide de géorequêtes, les champs de localisation que vous spécifiez `geoLocations` doivent correspondre aux champs de position auxquels vous passez`UpdateIndexingConfiguration`. En cas de non-concordance, l'indexation de la flotte envoie une erreur de type non concordant à. CloudWatch Le journal des erreurs contient le nom du champ, la valeur qui n'a pas pu être convertie et le nom d'objet de l'appareil.

Voici un exemple de journal des erreurs.

```
{
"timestamp": "2023-11-09 01:39:43.466",
    "logLevel": "ERROR",
    "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "IndexingGeoLocationFieldFailed",
    "thingName": "thing0",
    "failedGeolocationFields": [
        {
"Name": "attributeName1",
            "Value": "apple",
            "ExpectedType": "Geopoint"
        }
    ],
    "reason": "failed to index the field because it could not be converted to one of
 the expected geoLocation formats."
}
```

Pour de plus amples informations, veuillez consulter Indexation des données de localisation.

Dépannage des métriques de la flotte

Impossible de voir les points de données dans CloudWatch

Si vous parvenez à créer une métrique de flotte mais que vous ne pouvez pas y voir les points de données CloudWatch, il est probable que vous ne disposiez d'aucun élément répondant aux critères de chaîne de requête.

Consultez cet exemple de commande pour créer une métrique de parc :

```
aws iot create-fleet-metric --metric-name "example_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature>80" --period 60 --aggregation-field
 "attributes.temperature" --aggregation-type name=Statistics,values=count
```

Si aucun élément ne répond aux critères de chaîne de requête `--query-string "thingName:TempSensor* AND attributes.temperature>80"` :

- Vous pourrez ainsi créer une métrique de flotte et y afficher des points de données CloudWatch. `values=count` Les points de données de la valeur `count` sont toujours 0.
- Sinon `valuescount`, vous serez en mesure de créer une métrique de flotte, mais vous ne verrez pas la métrique de flotte dedans CloudWatch et il n'y aura aucun point de données à afficher CloudWatch.

AWS IoT Résolution des problèmes liés au catalogue des packages logiciels de gestion des appareils

Il s'agit de la section de dépannage du catalogue des packages logiciels de gestion des AWS IoT périphériques.

Messages d'erreur généraux relatifs au dépannage

Cette section répertorie les erreurs courantes observées tout au long du cycle de vie des versions des progiciels.

Erreurs **`HeadBucket`**

Les messages d'erreur suivants s'affichent lorsque vous appelez l'opération `HeadBucket` API ou la commande `head-bucket` CLI pour valider le compartiment Amazon S3 utilisé pour le téléchargement de fichiers lors d'un déploiement de tâches.

Pour plus d'informations sur l'utilisation d'un compartiment Amazon S3 pour le téléchargement de fichiers lors du déploiement d'une tâche, consultezURL présignée pour le téléchargement de fichiers.

`InvalidRoleException`

```
    "Permission denied when attempting to use role %s to access bucket %s."
```

`InvalidRequestException`

```
    "Cross region S3 bucket is not supported for presigned url upload placeholder"
```

```
InvalidRequestException
    "S3 bucket in job document presigned url upload placeholder not found"
```

```
InvalidRequestException
    "Given S3 bucket name is invalid."
```

```
InvalidRequestException
    "Provided S3 bucket is not valid: %s. Error: %s"
```

Amazon S3 GetObject

Le message d'erreur suivant apparaît lorsqu'un argument non valide est fourni, ce qui entraîne l'échec de l'opération d'`GetObject`API Amazon S3.

```
InvalidRequestException
    "Provided argument for presigned url is invalid"
```

Support de l'identifiant de version Amazon S3

Lorsque vous demandez l'accès à un compartiment Amazon S3 à l'aide du contrôle de version, assurez-vous d'inclure votre erreur `versionId` ou l'erreur ci-dessous peut s'afficher.

Pour plus d'informations sur les compartiments Amazon S3 utilisant le contrôle de version, consultez Utilisation du contrôle de version dans les compartiments Amazon S3

```
InvalidRequestException
    "VersionId not found when attempting to access s3 url"
```

Espaces réservés à l'intérieur d'une URL présignée pour le téléchargement de fichiers

Les messages d'erreur suivants apparaissent lorsque vous rencontrez des problèmes avec un espace réservé à l'intérieur d'une URL présignée utilisée pour télécharger des fichiers vers un compartiment Amazon S3 de destination lors du déploiement d'une tâche. Pour plus d'informations sur l'utilisation d'un compartiment Amazon S3 pour le téléchargement de fichiers lors du déploiement d'une tâche et sur ce qu'est un espace réservé local, consultez. URL présignée pour le téléchargement de fichiers

Le message d'erreur ci-dessous apparaît lorsque l'espace réservé local n'est pas reconnu.

```
InvalidJobDocumentException
    "Undefined placeholder, ${...}, inside of presign url upload parameter"
```

Le message d'erreur ci-dessous s'affiche lorsque vous essayez d'utiliser l'espace réservé local dans une URL présignée qui n'est pas destinée au téléchargement de fichiers.

```
InvalidJobDocumentException
    "Local placeholder, ${...}, is only valid inside of presign url upload"
```

URL Amazon S3 imbriquée de manière incorrecte

Le message d'erreur suivant s'affiche lorsque l'URL Amazon S3 est incorrectement imbriquée dans un autre espace réservé.

```
InvalidJobDocumentException
    "${aws:%s[...]} should not be the second layer pattern."
```

Version du package Artifact Nesting

Le message d'erreur suivant s'affiche lorsque l'URL présignée de l'artefact de la version du package est incorrectement imbriquée dans un autre espace réservé.

```
InvalidJobDocumentException
    "${aws:iot:package:[...]:artifact:s3-presigned-url} cannot be nested inside another
 placeholder."
```

Artifact de version du package manquant

Le message d'erreur suivant s'affiche lorsque l'artefact de version du package référencé est introuvable.

```
InvalidJobDocumentException
    "Package %s version %s does not have an associated artifact to generate an S3
 presigned url."
```

Packages logiciels et versions de packages substituables

Le message d'erreur suivant apparaît lorsque l'espace réservé au document de tâche pour le package logiciel et la version du package ne parvient pas à obtenir les valeurs valides souhaitées pour le déploiement du travail en raison de plusieurs packages logiciels et versions de package référencés dans le `destinationPackageVersions` paramètre ou dans l'onglet Version ARN de la page de détails de la version du package.

```
InvalidJobDocumentException
    "Cannot resolve empty package name and version name given multiple elements in
 destination package versions."
```

Utilisation d'un package logiciel et d'une version de package vides

Le message d'erreur suivant s'affiche lorsque vous essayez d'utiliser un package vide ou une version de package sans l'autre dans un document de travail.

```
InvalidJobDocumentException
    "Empty package name and version name have to be used in pair."
```

Utilisation nulle dans le document Job

Le message d'erreur suivant s'affiche lorsque vous essayez de spécifier `$null` une version de package dans le document de travail. `$null`ne peut être utilisé qu'à l'intérieur du `destinationPackageVersions` paramètre lors de l'utilisation de l'opération `CreateJob` API.

```
InvalidJobDocumentException
    "$null is not allowed to be referenced as a package version in job documents."
```

Tous les attributs d'une version de package

Le message d'erreur suivant s'affiche lorsque vous essayez d'utiliser tous les attributs d'une version de package et que vous l'entourez de texte supplémentaire ou d'espaces réservés.

Pour plus d'informations sur l'utilisation de tous les attributs dans une version de progiciel, voir Paramètres de substitution pour les AWS IoT tâches

```
InvalidJobDocumentException
    "The package version attribute placeholder for all attributes has to be a
 json value by itself and not appended with other strings or nested with other
 placeholders."
```

Limite d'espaces réservés locaux dans l'URL présignée pour le téléchargement de fichiers

Le message d'erreur suivant s'affiche lorsque vous dépassez la limite du nombre d'espaces réservés locaux utilisés dans une URL présignée pour le téléchargement de fichiers lors du déploiement d'une tâche.

Pour plus d'informations sur l'utilisation d'une URL présignée pour le téléchargement de fichiers lors du déploiement d'une tâche, voir URL présignée pour le téléchargement de fichiers

```
InvalidJobDocumentException
    "The occurrence of local placeholder %s within S3 presigned url upload placeholder
 exceeds limit of %d."
```

Espaces réservés locaux dans un compartiment Amazon S3

Le message d'erreur suivant s'affiche lorsque vous tentez de placer une URL d'espace réservé local dans le nom du compartiment Amazon S3 pour un espace réservé d'URL présigné utilisé pour le téléchargement de fichiers lors du déploiement d'une tâche.

Pour plus d'informations sur l'utilisation d'une URL présignée pour le téléchargement de fichiers lors du déploiement d'une tâche, voir URL présignée pour le téléchargement de fichiers

```
InvalidJobDocumentException
    "S3 bucket name in presigned url upload is not allowed to contain any placeholders"
```

Supports d'ouverture et de fermeture

Le message d'erreur suivant apparaît lorsque vous ajoutez un paramètre ou un espace réservé à un document de travail sans accolade «} ».

```
InvalidJobDocumentException
    "One or more parameters or placeholders are not terminated."
```

Rôle IAM avec URL présignée Amazon S3

Le message d'erreur suivant s'affiche lorsque vous tentez d'utiliser une URL présignée Amazon S3 dans un document de travail sans rôle IAM.

Pour plus d'informations sur Amazon S3 presigned URLs, consultez Travailler avec URLs Presigned.

```
InvalidRequestException
```

```
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document."
```

Rôle IAM avec URL présignée Amazon S3 pour Package Version Artifact

Le message d'erreur suivant s'affiche lorsque vous tentez d'utiliser une URL présignée Amazon S3 représentant un artefact de version de package dans un document de travail sans rôle IAM.

```
InvalidRequestException
    "presignedUrlConfig role ARN is required to generate an S3 presigned url in job
 document for package %s version %s artifact."
```

Messages d'erreur relatifs à la nomenclature logicielle

Cette section répertorie les erreurs courantes associées à une nomenclature logicielle (SBOM) liée à une version de package.

Validation des entrées pour la demande d'association SBOM

Le message d'erreur suivant s'affiche lors de l'utilisation de l'opération `AssociateSbomWithPackageVersion` API et le `s3Location` paramètre est nul.

```
InvalidRequestException "Associate request needs to include SBOM reference"
```

Pour plus d'informations sur le fonctionnement de l'`AssociateSbomWithPackageVersion`API, consultez AssociateSbomWithPackageVersion.

Erreurs de validation SBOM

Cette section répertorie les erreurs courantes observées lors de la validation initiale de la nomenclature logicielle (SBOM) lorsqu'elle est associée à une version de progiciel.

Le message d'erreur suivant s'affiche lors de l'utilisation de l'opération `AssociateSbomWithPackageVersion` API et `bucket` le `s3Location` paramètre contient la valeur null.

```
InvalidRequestException "S3 bucket name for SBOM cannot be null"
```

Le message d'erreur suivant s'affiche lorsque la `bucket` chaîne du `s3Location` paramètre pour l'opération `AssociateSbomWithPackageVersion` d'API est trop longue.

```
InvalidRequestException "S3 bucket name for SBOM is illegal. String length exceeds
 limit"
```

Le message d'erreur suivant s'affiche lorsque le `key` paramètre est nul.

```
InvalidRequestException "S3 key name for SBOM cannot be null"
```

Le message d'erreur suivant s'affiche lorsque la `key` chaîne du `s3Location` paramètre pour l'opération `AssociateSbomWithPackageVersion` d'API est trop longue.

```
InvalidRequestException "S3 key name for SBOM is illegal. String length exceeds limit"
```

Le message d'erreur suivant s'affiche lorsque la `version` chaîne du `s3Location` paramètre de l'opération `AssociateSbomWithPackageVersion` d'API est nulle.

```
InvalidRequestException "S3 object version for SBOM cannot be null"
```

Le message d'erreur suivant s'affiche lorsque la `version` chaîne du `s3Location` paramètre pour l'opération `AssociateSbomWithPackageVersion` d'API est trop longue.

```
InvalidRequestException "S3 object version for SBOM is illegal. String length exceeds
 limit"
```

Le message d'erreur suivant s'affiche lorsque la taille du fichier d'archive zip SBOM stocké dans le compartiment Amazon S3 est trop grande.

```
InvalidRequestException "S3 object file size exceeds limit"
```

Le message d'erreur suivant s'affiche lorsque vous utilisez l'opération `AssociateSbomWithPackageVersion` API et que le nombre actuel de validations SBOM en cours est déjà atteint la limite maximale.

```
LimitExceededException "Too many ongoing SBOM validation workflows. Please wait and
 retry"
```

Problèmes d'accès liés au fichier SBOM dans le compartiment Amazon S3

Le message d'erreur suivant s'affiche lorsqu'une autre entité ne parvient pas à accéder au compartiment Amazon S3 parce que celui-ci n'existe pas ou que les autorisations appropriées n'ont pas été accordées pour accéder au compartiment Amazon S3.

Pour plus d'informations sur la politique d'autorisation requise pour accéder au compartiment Amazon S3, consultezStockage de la nomenclature logicielle.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket exists and S3 permission is granted."
```

Le message d'erreur suivant apparaît lorsqu'une autre entité ne parvient pas à accéder au fichier d'archive zip SBOM dans le `key` paramètre parce que le compartiment Amazon S3 n'existe pas ou que les autorisations appropriées n'ont pas été accordées pour accéder au contenu stocké dans le compartiment Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the key
 exists and S3 permission is granted."
```

Le message d'erreur suivant s'affiche lorsqu'une autre entité ne parvient pas à accéder au compartiment Amazon S3 parce que le compartiment, la clé et l'ID de version n'existent pas ou parce que les autorisations appropriées n'ont pas été accordées pour accéder au compartiment Amazon S3. En outre, ce message d'erreur peut s'afficher si les autorisations accordées sont insuffisantes pour accéder au fichier d'archive zip SBOM dans le compartiment Amazon S3.

```
InvalidRequestException "SBOM not accessible by the service. Please make sure the
 bucket/key/version exists and S3 permission is granted."
```

Le message d'erreur suivant s'affiche lorsqu'une autre entité ne parvient pas à accéder au compartiment Amazon S3 parce que le compartiment est situé dans une autre région.

```
InvalidRequestException "Cross-region S3 bucket for %s is not supported."
```

Le message d'erreur suivant s'affiche lorsqu'une autre entité ne parvient pas à accéder au compartiment Amazon S3 en raison de l'`bucket`orthographe incorrecte des `version` paramètres lors de l'utilisation de l'`AssociateSbomWithPackageVersion`API. `key`

```
InvalidRequestException "Please make sure SBOM S3 bucket name/key length/version is
 valid"
```

AWS IoT Guide de dépannage de Device Advisor

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

Général

Q : Puis-je exécuter plusieurs suites de tests en parallèle ?

A : Oui. Device Advisor prend désormais en charge l'exécution de plusieurs suites de tests sur différents appareils à l'aide d'un point de terminaison au niveau de l'appareil. Si vous utilisez le point de terminaison au niveau du compte, vous pouvez exécuter une suite à la fois car un point de terminaison Device Advisor est disponible par compte. Pour de plus amples informations, veuillez consulter configuration de votre appareil

Q : J'ai vu sur mon appareil que la connexion TLS avait été refusée par Device Advisor. Cela est-il prévu ?

A : Oui. Device Advisor refuse la connexion TLS avant et après chaque essai. Nous recommandons aux utilisateurs de mettre en œuvre un mécanisme de nouvelle tentative sur l'appareil afin de bénéficier d'une expérience de test entièrement automatisée avec Device Advisor. Si vous exécutez une suite de tests comportant plusieurs scénarios de test, par exemple TLS connect, MQTT connect et MQTT publish, nous vous recommandons de créer un mécanisme adapté à votre appareil. Le mécanisme peut essayer de se connecter à notre point de terminaison de test toutes les 5 secondes pendant une minute à deux. De cette manière, vous pouvez exécuter plusieurs cas de test en séquence de manière automatisée.

Q : Puis-je obtenir un historique des appels API de Device Advisor effectués sur mon compte à des fins d'analyse de la sécurité et de dépannage opérationnel ?

A : Oui. Pour recevoir l'historique des appels d'API Device Advisor effectués sur votre compte, il vous suffit d'activer la console de AWS IoT gestion et de filtrer la source d'événement à utiliser`iotdeviceadvisor.amazonaws.com`. CloudTrail

Q : Comment puis-je consulter les connexions à Device Advisor CloudWatch ?

R : Les journaux générés lors de l'exécution d'une suite de tests sont téléchargés CloudWatch si vous ajoutez la politique requise (par exemple, CloudWatchFullAccess) à votre rôle de service (voirConfiguration). S'il existe au moins un cas de test dans la suite de tests, un groupe de

journaux « aws/iot/deviceadvisor /$ testSuiteId » est créé avec deux flux de journaux. L'un des flux est le « $ testRunId » et inclut les journaux des actions effectuées avant et après l'exécution des scénarios de test dans votre suite de tests, telles que les étapes de configuration et de nettoyage. L'autre flux de log est « $ suiteRunId _$ »testRunId, qui est spécifique à une suite de tests exécutée. Les événements sont envoyés depuis des appareils et AWS IoT Core seront enregistrés dans ce flux de journal.

Q : Quel est l'objectif du rôle d'autorisation de l'appareil ?

R : Device Advisor se trouve entre votre appareil de test et permet AWS IoT Core de simuler des scénarios de test. Il accepte les connexions et les messages de vos appareils de test et les transmet AWS IoT Core en assumant le rôle d'autorisation de votre appareil et en établissant une connexion en votre nom. Il est important de vous assurer que les autorisations relatives aux rôles de l'appareil sont les mêmes que celles du certificat que vous utilisez pour exécuter des tests. AWS IoT les politiques de certification ne sont pas appliquées lorsque Device Advisor établit une connexion en votre AWS IoT Core nom en utilisant le rôle d'autorisation de l'appareil. Toutefois, les autorisations associées au rôle d'autorisation de l'appareil que vous avez défini sont appliquées.

Q : Dans quelles régions Device Advisor est-il pris en charge ?

Device Advisor est pris en charge dans les régions us-west-2 et eu-west-1.

Q : Pourquoi est-ce que je constate des résultats incohérents ?

R : L'une des principales causes d'incohérence des résultats est la définition d'un test `EXECUTION_TIMEOUT` à une valeur trop faible. Pour plus d'informations sur les `EXECUTION_TIMEOUT` valeurs recommandées et par défaut, consultez les scénarios de test de Device Advisor.

Q : Quel est le protocole MQTT pris en charge par Device Advisor ?

R : Device Advisor prend en charge la version 3.1.1 de MQTT avec les certificats clients X509.

Q : Et si mon scénario de test a échoué avec un message d'expiration du délai d'exécution alors que j'ai essayé de connecter mon appareil au point de terminaison du test ?

R : Validez toutes les étapes de la section Créer un rôle IAM à utiliser comme rôle sur votre appareil. Si le test échoue toujours, il se peut que l'appareil n'envoie pas l'extension SNI (Server Name Indication) correcte, requise pour que Device Advisor fonctionne. La valeur SNI correcte est l'adresse du point de terminaison renvoyée lorsque vous suivez la section Configurer votre appareil. AWS IoT exige également que les appareils envoient l'extension SNI (Server Name

Indication) au protocole TLS (Transport Layer Security). Pour plus d'informations, consultez la section Sécurité du transport dans AWS IoT.

Q : Ma connexion MQTT échoue avec une erreur « libaws-c-mqtt : AWS_ERROR _MQTT_UNEXPECTED_HANGUP » (ou) la connexion MQTT de mon appareil est automatiquement déconnectée du point de terminaison Device Advisor. Comment résoudre cette erreur ?

R : Ce code d'erreur particulier et les déconnexions inattendues peuvent être provoqués par de nombreux facteurs différents, mais ils sont probablement liés au rôle de l'appareil attaché à l'appareil. Les points de contrôle ci-dessous (par ordre de priorité) permettront de résoudre ce problème.

- Le rôle de périphérique attaché à l'appareil doit disposer des autorisations IAM minimales requises pour exécuter les tests. Device Advisor utilisera le rôle de périphérique attaché pour effectuer des actions AWS IoT MQTT au nom du périphérique de test. Si les autorisations requises sont absentes, l'`AWS_ERROR_MQTT_UNEXPECTED_HANGUP`erreur s'affichera ou des déconnexions inattendues se produiront pendant que l'appareil tente de se connecter au point de terminaison Device Advisor. Par exemple, si vous avez choisi d'exécuter le scénario de test MQTT Publish, les actions Connect et Publish doivent être incluses dans le rôle avec le sujet correspondant ClientId (vous pouvez fournir plusieurs valeurs en utilisant des virgules pour séparer les valeurs, et vous pouvez fournir des valeurs de préfixe à l'aide d'un caractère générique (*). Par exemple, pour accorder l'autorisation de publier sur n'importe quel sujet commençant par `TestTopic`, vous pouvez fournir « `TestTopic*` » comme valeur de ressource. Voici quelques exemples de politiques.
- Incompatibilité entre les valeurs définies dans le rôle de l'appareil pour vos types de ressources et les valeurs réelles utilisées dans le code. Par exemple : une incompatibilité est ClientId définie dans le rôle et le code réellement ClientId utilisé dans le code de votre appareil. Les valeurs telles que ClientId Topic et le code TopicFilter doivent être identiques dans le rôle et le code de l'appareil.
- Le certificat d'appareil associé à votre appareil doit être actif et être associé à une politique comportant les autorisations d'action requises pour les ressources. Notez que la politique de certification des appareils accorde ou refuse l'accès aux AWS IoT ressources et aux opérations du plan de AWS IoT Core données. Device Advisor exige que vous disposiez d'un certificat d'appareil actif attaché à votre appareil qui accorde les autorisations d'action utilisées lors d'un scénario de test.

AWS IoT erreurs

Aidez-nous à améliorer ce sujet

Dites-nous ce qui pourrait contribuer à l'améliorer

Cette section répertorie les codes d'erreur envoyés par AWS IoT.

Codes d'erreur de l'agent de messages

Code d’erreur	Description de l'erreur.
400	Demande erronée.
401	Accès non autorisé.
403	Accès interdit.
426	Mise à niveau requise.
503	Service non disponible.

Identités et des codes d'erreur de sécurité

Code d’erreur	Description de l'erreur.
401	Accès non autorisé.

Codes d'erreur Device Shadow

Code d’erreur	Description de l'erreur.
400	Demande erronée.
401	Accès non autorisé.
403	Accès interdit.
404	Introuvable.

Code d’erreur	Description de l'erreur.
409	Conflit.
413	Demande trop longue.
422	Impossible de traiter la demande.
429	Nombre de demandes trop élevé.
500	Erreur interne.
503	Service non disponible.

AWS IoT Appareil SDKs, appareil mobile SDKs et client AWS IoT d'appareil

Cette page résume l' AWS IoT appareil SDKs, les bibliothèques open source, les guides du développeur, les exemples d'applications et les guides de portage pour vous aider à créer des solutions IoT innovantes avec AWS IoT les plateformes matérielles de votre choix.

Ils SDKs sont destinés à être utilisés sur votre appareil IoT. Si vous développez une application IoT destinée à être utilisée sur un appareil mobile, consultez le AWS Applications mobiles SDKs. Si vous développez une application IoT ou un programme côté serveur, consultez le AWS SDKs.

AWS IoT Device SDKs

L' AWS IoT appareil SDKs comprend des bibliothèques open source, des guides de développement avec des exemples et des guides de portage afin que vous puissiez créer des produits ou des solutions IoT innovants sur les plateformes matérielles de votre choix.

> **Note**
>
> L' AWS IoT appareil SDKs a publié un client MQTT 5. L' AWS IoT appareil SDKs ne prend pas en charge l'utilisation du protocole TLS 1.3 sur macOS.

Ils vous SDKs aident à connecter vos appareils IoT à AWS IoT l'aide des protocoles MQTT et WSS.

C++

AWS IoT SDK pour appareils C++

Le SDK AWS IoT C++ Device permet aux développeurs de créer des applications connectées à l'aide AWS du AWS IoT APIs. Ce kit SDK a été conçu en particulier pour les appareils qui ne sont pas limités en ressources et qui nécessitent des fonctions avancées, telles que la mise en file d'attente des messages, la prise en charge du multithreading et les dernières fonctions de langue. Pour plus d’informations, consultez les ressources suivantes :

- AWS IoT SDK de l'appareil C ++ v2 activé GitHub
- AWS IoT Readme du SDK C++ v2 de l'appareil
- AWS IoT Exemples du SDK C++ v2 pour appareils

- AWS IoT Documentation de l'API C++ v2 du SDK pour appareils

Python

AWS IoT SDK de périphérique pour Python

Le AWS IoT Device SDK for Python permet aux développeurs d'écrire des scripts Python afin d'utiliser leurs appareils pour accéder à la AWS IoT plateforme via MQTT ou MQTT via le protocole. WebSocket En connectant leurs appareils à AWS IoT, les utilisateurs peuvent travailler en toute sécurité avec le courtier de messages, les règles et les ombres fournis par AWS IoT et avec d'autres AWS services tels que AWS Lambda Kinesis et Amazon S3, etc.

- AWS IoT SDK de périphérique pour Python v2 sur GitHub
- AWS IoT SDK du périphérique pour Python v2 Readme
- AWS IoT Exemples de SDK de périphérique pour Python v2
- AWS IoT Documentation de l'API du SDK de l'appareil pour Python v2

JavaScript

AWS IoT SDK de périphérique pour JavaScript

Le package aws-iot-device-sdk .js permet aux développeurs d'écrire des JavaScript applications qui accèdent à AWS IoT l'aide de MQTT ou MQTT via le protocole. WebSocket Il peut être utilisé dans des environnements Node.js et des applications de navigateur. Pour plus d'informations, consultez les ressources suivantes :

- AWS IoT SDK de l'appareil pour la JavaScript version 2 sur GitHub
- AWS IoT SDK du périphérique pour Readme JavaScript v2
- AWS IoT Exemples de SDK du périphérique pour la JavaScript version v2
- AWS IoT Documentation de l'API du SDK de l'appareil pour la JavaScript version v2

Java

AWS IoT SDK du périphérique pour Java

Le AWS IoT Device SDK for Java permet aux développeurs Java d'accéder à AWS IoT la plateforme via MQTT ou MQTT via le protocole. WebSocket Le kit SDK est intégré à la prise en charge des shadows. Vous pouvez accéder au service Shadows à l'aide des méthodes HTTP,

notamment GET, UPDATE et DELETE. Le kit SDK prend également en charge un modèle d'accès aux shadows simplifié, qui permet aux développeurs d'échanger des données avec des shadows en utilisant uniquement des méthodes getter et setter, sans avoir à sérialiser ou désérialiser des documents JSON.

> **Note**
>
> Le AWS IoT Device SDK for Java v2 prend désormais en charge le développement Android. Pour plus d'informations, consultez AWS IoT Device SDK for Android.

Pour plus d’informations, consultez les ressources suivantes :

- AWS IoT SDK de périphérique pour Java v2 activé GitHub
- AWS IoT SDK du périphérique pour Java v2 Readme
- AWS IoT Exemples de SDK pour appareils pour Java v2
- AWS IoT Documentation de l'API Device SDK for Java v2

Swift

AWS IoT SDK de périphérique pour Swift

Le SDK AWS IoT Device pour Swift permet aux développeurs Swift de créer des AWS IoT applications pour les plateformes Linux et Apple macOS, iOS et tvOS à l'aide du protocole MQTT 5.

Pour plus d’informations, consultez les ressources suivantes :

- AWS IoT SDK de l'appareil pour Swift on GitHub
- AWS IoT SDK de périphérique pour Swift Readme
- AWS IoT SDK du périphérique pour Swift Samples

AWS IoT SDK de périphérique pour Embedded C

> **Note**
>
> Ce SDK est destiné à être utilisé par des développeurs de logiciels embarqués expérimentés.

Le Kit SDK des appareils AWS IoT pour Embedded C (C-SDK) est une collection de fichiers source C sous licence open source du MIT qui peuvent être utilisés dans des applications intégrées pour connecter en toute sécurité des appareils IoT à. AWS IoT Core Il inclut un client MQTT, un analyseur JSON, AWS IoT Device Shadow, AWS IoT Jobs, AWS IoT Fleet Provisioning et des bibliothèques. AWS IoT Device Defender Ce SDK est distribué sous forme source et peut être intégré au micrologiciel client avec le code d'application, d'autres bibliothèques et un système d'exploitation (OS) de votre choix.

Kit SDK des appareils AWS IoT pour Embedded C Il est généralement destiné aux appareils aux ressources limitées qui nécessitent un environnement d'exécution en langage C optimisé. Vous pouvez utiliser le SDK sur n'importe quel système d'exploitation et l'héberger sur n'importe quel type de processeur (par exemple, MCUs et MPUs).

Pour plus d’informations, consultez les ressources suivantes :

- AWS IoT SDK du périphérique pour Embedded C on GitHub
- AWS IoT SDK du périphérique pour Embedded C Readme
- AWS IoT SDK de périphérique pour les échantillons C intégrés

AWS Applications mobiles SDKs

The AWS Mobile SDKs fournit aux développeurs d'applications mobiles une assistance spécifique à la plate-forme pour les services AWS IoT Core , APIs la communication entre appareils IoT à l'aide de MQTT et les APIs autres services. AWS

Android

AWS Mobile SDK for Android

AWS Mobile SDK for Android Il contient une bibliothèque, des exemples et de la documentation permettant aux développeurs de créer des applications mobiles connectées à l'aide de AWS. Ce SDK inclut également la prise en charge des communications entre appareils MQTT et APIs de l' AWS IoT Core appel des services. Pour plus d’informations, consultez les ressources suivantes :

- AWS Mobile SDK for Android sur GitHub
- AWS Mobile SDK for Android Readme
- AWS Mobile SDK for Android Exemples
- AWS Mobile SDK for Android Référence d'API

- AWSIoTClient Documentation de référence sur les classes

iOS

AWS Mobile SDK for iOS

AWS Mobile SDK for iOS Il s'agit d'un kit de développement logiciel open source, distribué sous licence Apache Open Source. AWS Mobile SDK for iOS Il fournit une bibliothèque, des exemples de code et de la documentation pour aider les développeurs à créer des applications mobiles connectées à l'aide de AWS. Ce SDK inclut également la prise en charge des communications entre appareils MQTT et APIs de l' AWS IoT Core appel des services. Pour plus d'informations, consultez les ressources suivantes :

- AWS Mobile SDK for iOS sur GitHub
- AWS Mobile SDK for iOS Readme
- AWS Mobile SDK for iOS Exemples
- AWSIoDocuments de référence sur la classe T dans le AWS Mobile SDK for iOS

AWS IoT Client de l'appareil

Le AWS IoT Device Client fournit du code pour aider votre appareil à se connecter AWS IoT, à effectuer des tâches de provisionnement du parc, à prendre en charge les politiques de sécurité des appareils, à se connecter à l'aide d'un tunneling sécurisé et à traiter les tâches sur votre appareil. Vous pouvez installer ce logiciel sur votre appareil pour gérer ces tâches de routine afin de vous concentrer sur votre solution spécifique.

> **Note**
>
> Le AWS IoT Device Client fonctionne avec des appareils IoT basés sur des microprocesseurs dotés de processeurs x86_64 ou ARM et de systèmes d'exploitation Linux courants.

C++

AWS IoT Client de l'appareil

Pour plus d'informations sur le client de AWS IoT périphérique en C++, consultez les rubriques suivantes :

- AWS IoT Client de périphérique dans le code source C++ sur GitHub
- AWS IoT Client de périphérique dans C++ Readme

SDKs Versions antérieures AWS IoT de l'appareil

Il s'agit de versions antérieures de AWS IoT Device SDKs qui ont été remplacées par les nouvelles versions répertoriées ci-dessus. Ils ne SDKs reçoivent que des mises à jour de maintenance et de sécurité. Ils ne seront pas mis à jour pour inclure de nouvelles fonctionnalités et ne doivent pas être utilisés sur de nouveaux projets.

- AWS IoT SDK de périphérique C++ activé GitHub
- AWS IoT Readme du SDK pour appareils C++
- AWS IoT SDK de périphérique pour Python v1 activé GitHub
- AWS IoT SDK du périphérique pour Python v1 Readme
- AWS IoT SDK de l'appareil pour Java activé GitHub
- AWS IoT SDK du périphérique pour Java Readme
- AWS IoT SDK de l'appareil pour JavaScript on GitHub
- AWS IoT SDK de périphérique pour Readme JavaScript
- SDK Arduino Yún activé GitHub
- Kit SDK Arduino Yún – Readme

Exemples de code à AWS IoT utiliser AWS SDKs

Les exemples de code suivants montrent comment utiliser AWS IoT un kit de développement AWS logiciel (SDK).

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Mise en route

Bonjour AWS IoT

Les exemples de code suivants montrent comment démarrer avec AWS IoT.

C++

SDK pour C++

Code pour le CMake fichier CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code du fichier source hello_iot.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
```

```
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
 iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
 listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() <<
 std::endl;
                break;
            }
        } while (!nextToken.empty());
```

```
        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Pour plus de détails sur l'API, voir ListThings dans le manuel de référence des AWS SDK pour C++ API.

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
```

```
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Pour plus de détails sur l'API, voir ListThings dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
```

```
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, voir ListThings dans le AWS SDK pour la référence de l'API Kotlin.

Exemples de code

- Exemples de base pour AWS IoT l'utilisation AWS SDKs
  - Bonjour AWS IoT
  - Apprenez les bases AWS IoT d'un AWS SDK
  - Actions d' AWS IoT utilisation AWS SDKs
    - Utilisation AttachThingPrincipal avec un AWS SDK ou une CLI
    - Utilisation CreateKeysAndCertificate avec un AWS SDK ou une CLI
    - Utilisation CreateThing avec un AWS SDK ou une CLI
    - Utilisation CreateTopicRule avec un AWS SDK ou une CLI
    - Utilisation DeleteCertificate avec un AWS SDK ou une CLI
    - Utilisation DeleteThing avec un AWS SDK ou une CLI
    - Utilisation DeleteTopicRule avec un AWS SDK ou une CLI

- Utilisation DescribeEndpoint avec un AWS SDK ou une CLI
- Utilisation DescribeThing avec un AWS SDK ou une CLI
- Utilisation DetachThingPrincipal avec un AWS SDK ou une CLI
- Utilisation ListCertificates avec un AWS SDK ou une CLI
- Utilisation ListThings avec un AWS SDK ou une CLI
- Utilisation SearchIndex avec un AWS SDK ou une CLI
- Utilisation UpdateIndexingConfiguration avec un AWS SDK ou une CLI
- Utilisation UpdateThing avec un AWS SDK ou une CLI

Exemples de base pour AWS IoT l'utilisation AWS SDKs

Les exemples de code suivants montrent comment utiliser les principes de base de AWS IoT with AWS SDKs.

Exemples

- Bonjour AWS IoT
- Apprenez les bases AWS IoT d'un AWS SDK
- Actions d' AWS IoT utilisation AWS SDKs
  - Utilisation AttachThingPrincipal avec un AWS SDK ou une CLI
  - Utilisation CreateKeysAndCertificate avec un AWS SDK ou une CLI
  - Utilisation CreateThing avec un AWS SDK ou une CLI
  - Utilisation CreateTopicRule avec un AWS SDK ou une CLI
  - Utilisation DeleteCertificate avec un AWS SDK ou une CLI
  - Utilisation DeleteThing avec un AWS SDK ou une CLI
  - Utilisation DeleteTopicRule avec un AWS SDK ou une CLI
  - Utilisation DescribeEndpoint avec un AWS SDK ou une CLI
  - Utilisation DescribeThing avec un AWS SDK ou une CLI
  - Utilisation DetachThingPrincipal avec un AWS SDK ou une CLI
  - Utilisation ListCertificates avec un AWS SDK ou une CLI
  - Utilisation ListThings avec un AWS SDK ou une CLI
  - Utilisation SearchIndex avec un AWS SDK ou une CLI

- Utilisation UpdateIndexingConfiguration avec un AWS SDK ou une CLI
- Utilisation UpdateThing avec un AWS SDK ou une CLI

Bonjour AWS IoT

Les exemples de code suivants montrent comment démarrer avec AWS IoT.

C++

Kit de développement logiciel (SDK) for C++

Code pour le CMake fichier CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code du fichier source hello_iot.cpp.

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 *  A "Hello IoT" starter application which initializes an AWS IoT client and
 *  lists the AWS IoT topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_iot'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
 iotClient.ListThings(
                    listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
 listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                          << listThingsOutcome.GetError().GetMessage() <<
 std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Pour plus de détails sur l'API, voir ListThings dans le manuel de référence des AWS SDK pour C++ API.

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import software.amazon.awssdk.services.iot.paginators.ListThingsIterable;

import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
 Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }

    public static void listAllThings(IotClient iotClient) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
```

```
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- Pour plus de détails sur l'API, voir ListThings dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listThings(thingsRequest)
        val thingList = response.things
        if (thingList != null) {
            for (attribute in thingList) {
```

```
                println("Thing name ${attribute.thingName}")
                println("Thing ARN: ${attribute.thingArn}")
            }
        }
    }
}
```

- Pour plus de détails sur l'API, voir ListThings dans le AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Apprenez les bases AWS IoT d'un AWS SDK

Les exemples de code suivants montrent comment :

- Créez un AWS IoT objet.
- Générez un certificat d'appareil.
- Mettez à jour un AWS IoT objet avec des attributs.
- Renvoie un point de terminaison unique.
- Répertoriez vos AWS IoT certificats.
- Créez une AWS IoT ombre.
- Rédigez les informations sur l'état.
- Crée une règle.
- Dressez la liste de vos règles.
- Recherchez des objets en utilisant le nom de l'objet.
- Supprimer un AWS IoT objet.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Créez n'importe AWS IoT quoi.

```
    Aws::String thingName = askQuestion("Enter a thing name: ");

    if (!createThing(thingName, clientConfiguration)) {
        std::cerr << "Exiting because createThing failed." << std::endl;
        cleanup("", "", "", "", "", false, clientConfiguration);
        return false;
    }
```

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
    }

    return outcome.IsSuccess();
}
```

Générez et joignez un certificat d'appareil.

```
    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing?
(y/n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
                "Would you like to save the certificate and keys to file? (y/n)
")) {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                      << outputFolder << std::endl;
        }

        if (!createKeysAndCertificate(outputFolder, certificateARN,
certificateID,
                                      clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                      << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
                  << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName,
clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
            cleanup(thingName, certificateARN, certificateID, "", "",
                    false,
                    clientConfiguration);
```

```
            return false;
        }
    }
```

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;
```

```
            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Effectuez diverses opérations sur la AWS IoT chose.

```
    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
 "v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
   printAsterisksLine();

   std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
   std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
   << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

   askQuestion("Press Enter to continue:", alwaysTrueTest);

   Aws::String endpoint;
   if (!describeEndpoint(endpoint, clientConfiguration)) {
       std::cerr << "Exiting because getEndpoint failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }
   std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
   printAsterisksLine();

   std::cout << "Now the certificates in your account will be listed." <<
std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);

   if (!listCertificates(clientConfiguration)) {
       std::cerr << "Exiting because listCertificates failed." << std::endl;
       cleanup(thingName, certificateARN, certificateID, "", "", false,
               clientConfiguration);
       return false;
   }

   printAsterisksLine();

   std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
   std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\n"
   << "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
   << "the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow." << std::endl;
   askQuestion("Press Enter to continue:", alwaysTrueTest);
```

```
    if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
        std::cerr << "Exiting because updateThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now, the state information for the shadow will be retrieved.\n"
 << std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String shadowState;
    if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
        std::cerr << "Exiting because getThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to
 access data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation
 template." << std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
 =createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
 std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
                clientConfiguration);
        return false;
    }
```

```
    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."
 << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    Aws::String topicArn = topicArnIter->second;
    Aws::String roleArn = roleArnIter->second;
    Aws::String sqlStatement = "SELECT * FROM '";
    sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
    sqlStatement += "'";

    printAsterisksLine();

    std::cout << "Now a rule will be created.\n" << std::endl;
    std::cout << "Rules are an administrator-level action. Any user who has
 permission\n"
                 << "to create rules will be able to access data processed by the
 rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
 std::endl;
    std::cout << "For more information on IoT SQL, see https://
docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" <<
 std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
 clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
```

```
    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }

    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/
latest/developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
 account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
       << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/
latest/developerguide/managing-index.html" << std::endl;
    if (askYesNoQuestion("Do you want to enable thing indexing in your account?
 (y/n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

 thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGI

 thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnec
        // The ThingGroupIndexingConfiguration object is ignored if not set.
        Aws::IoT::Model::ThingGroupIndexingConfiguration
 thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
 thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
 std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                    ruleName, false,
                    clientConfiguration);
            return false;
```

```
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
                false,
                clientConfiguration);
        return false;
    }
```

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;
```

```
    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param document: The state information, in JSON format.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
 iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
 std::make_shared<std::stringstream>(
            document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
 iotDataPlaneClient.UpdateThingShadow(
            updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param documentResult: String to receive the state information, in JSON format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                 Aws::String &documentResult,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
```

```
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});
```

```
    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome =
 iotClient.ListTopicRules(
                request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
 outcome.GetResult();
            allRules.insert(allRules.end(),
                            result.GetRules().cbegin(),
                            result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
```

```
        else {
            std::cerr << "ListTopicRules error: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
              << std::endl;
    for (auto &rule: allRules) {
        std::cout << "  Rule name: " << rule.GetRuleName() << ", rule ARN: "
                  << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

Nettoyez les ressources.

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
 &certificateARN,
                     const Aws::String &certificateID, const Aws::String
 &stackName,
                     const Aws::String &ruleName, bool askForConfirmation,
                     const Aws::Client::ClientConfiguration &clientConfiguration)
 {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
                              askYesNoQuestion("Delete the rule '" + ruleName +
                                               "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
```

```
    }

    Aws::CloudFormation::CloudFormationClient
 cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
                               askYesNoQuestion(
                                       "Delete the CloudFormation stack '" +
 stackName +
                                       "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
                                    askYesNoQuestion("Delete the certificate '" +
                                                     certificateARN + "'? (y/n)
 "))) {
        result &= detachThingPrincipal(certificateARN, thingName,
 clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName
 +
                                                "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}
```

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
```

```
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
           detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
           request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Exécutez un scénario interactif illustrant AWS IoT les fonctionnalités.

```
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
```

```
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
 with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series
of steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality
for creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS
IoT capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);

        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service
that can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName
+"? (y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
```

```
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a
pivotal advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon
Web Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a
virtual representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For
example, you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON
format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate
for " +thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
```

```
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Une classe wrapper pour les méthodes du AWS IoT SDK.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotAsyncClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.Certificate;
import
 software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
```

```
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import
 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
```

```
        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))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        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))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
```

```
        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT
certificate.
     * If the request is successful, it prints the certificate details and
returns the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
     *
     * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
     * If the request is successful, it prints a confirmation message and
additional information about the Thing.
     * If an exception occurs, it prints the error message.
     */
    public void attachCertificateToThing(String thingName, String certificateArn)
{
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing
successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
```

```
                        attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });
```

```
        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

```
    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
 getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
```

```
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a
Thing's shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();
```

```
        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
```

```
                List<TopicRuleListItem> ruleList =
listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
```

```
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
```

```
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed
from " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully
deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
```

```
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") +
 1);
    }
}
```

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteStream
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
```

```
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development
 environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-
getting-started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
 IOT.
            snsAction  - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
```

```
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an
IoT thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that
can be associated with a physical device.
        """.trimIndent(),
    )
    // Prompt the user for input.
    print("Enter thing name: ")
    thingName = scanner.nextLine()
    createIoTThing(thingName)
    describeThing(thingName)
    println(DASHES)

    println(DASHES)
    println("2. Generate a device certificate.")
    println(
        """
```

```
        A device certificate performs a role in securing the communication
between devices (things) and the AWS IoT platform.
        """.trimIndent(),
    )

    print("Do you want to create a certificate for $thingName? (y/n)")
    val certAns = scanner.nextLine()
    var certificateArn: String? = ""
    if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
        certificateArn = createCertificate()
        println("Attach the certificate to the AWS IoT thing.")
        attachCertificateToThing(thingName, certificateArn)
    } else {
        println("A device certificate was not created.")
    }
    println(DASHES)

    println(DASHES)
    println("3. Update an AWS IoT thing with Attributes.")
    println(
        """
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateThing(thingName)
    println(DASHES)

    println(DASHES)
    println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
    println(
        """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
```

```
    val endpointUrl = describeEndpoint()
    println(DASHES)

    println(DASHES)
    println("5. List your AWS IoT certificates")
    print("Press Enter to continue...")
    scanner.nextLine()
    if (certificateArn!!.isNotEmpty()) {
        listCertificates()
    } else {
        println("You did not create a certificates. Skipping this step.")
    }
    println(DASHES)

    println(DASHES)
    println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
    println(
        """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
    print("Press Enter to continue...")
    scanner.nextLine()
    updateShawdowThing(thingName)
    println(DASHES)

    println(DASHES)
    println("7. Write out the state information, in JSON format.")
    print("Press Enter to continue...")
    scanner.nextLine()
    getPayload(thingName)
    println(DASHES)

    println(DASHES)
    println("8. Creates a rule")
    println(
        """
```

```
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """.trimIndent(),
    )
    print("Enter Rule name: ")
    val ruleName = scanner.nextLine()
    createIoTRule(roleARN, ruleName, snsAction)
    println(DASHES)

    println(DASHES)
    println("9. List your rules.")
    print("Press Enter to continue...")
    scanner.nextLine()
    listIoTRules()
    println(DASHES)

    println(DASHES)
    println("10. Search things using the name.")
    print("Press Enter to continue...")
    scanner.nextLine()
    val queryString = "thingName:$thingName"
    searchThings(queryString)
    println(DASHES)

    println(DASHES)
    if (certificateArn.length > 0) {
        print("Do you want to detach and delete the certificate for $thingName?
(y/n)")
        val delAns = scanner.nextLine()
        if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
            println("11. You selected to detach amd delete the certificate.")
            print("Press Enter to continue...")
            scanner.nextLine()
            detachThingPrincipal(thingName, certificateArn)
            deleteCertificate(certificateArn)
        } else {
            println("11. You selected not to delete the certificate.")
        }
    } else {
        println("11. You did not create a certificate so there is nothing to
delete.")
    }
```

```
    println(DASHES)

    println(DASHES)
    println("12. Delete the AWS IoT thing.")
    print("Do you want to delete the IoT thing? (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
 true)) {
        deleteIoTThing(thingName)
    } else {
        println("The IoT thing was not deleted.")
    }
    println(DASHES)

    println(DASHES)
    println("The AWS IoT workflow has successfully completed.")
    println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
```

```
    val arnParts = certificateArn.split(":".toRegex()).dropLastWhile
 { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
```

```
        val listTopicRulesResponse =
 iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("--------------")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
```

```
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        val getThingShadowResponse =
 iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
```

```
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShawdowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
```

```
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity
\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient.fromEnvironment { region = "us-east-1" }.use
 { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
```

```
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}

suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Actions d' AWS IoT utilisation AWS SDKs

Les exemples de code suivants montrent comment effectuer des AWS IoT actions individuelles avec AWS SDKs. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, veuillez consulter la AWS IoT Référence d'API.

Exemples

- Utilisation AttachThingPrincipal avec un AWS SDK ou une CLI
- Utilisation CreateKeysAndCertificate avec un AWS SDK ou une CLI
- Utilisation CreateThing avec un AWS SDK ou une CLI
- Utilisation CreateTopicRule avec un AWS SDK ou une CLI
- Utilisation DeleteCertificate avec un AWS SDK ou une CLI
- Utilisation DeleteThing avec un AWS SDK ou une CLI
- Utilisation DeleteTopicRule avec un AWS SDK ou une CLI
- Utilisation DescribeEndpoint avec un AWS SDK ou une CLI
- Utilisation DescribeThing avec un AWS SDK ou une CLI
- Utilisation DetachThingPrincipal avec un AWS SDK ou une CLI
- Utilisation ListCertificates avec un AWS SDK ou une CLI
- Utilisation ListThings avec un AWS SDK ou une CLI
- Utilisation SearchIndex avec un AWS SDK ou une CLI
- Utilisation UpdateIndexingConfiguration avec un AWS SDK ou une CLI
- Utilisation UpdateThing avec un AWS SDK ou une CLI

Utilisation **`AttachThingPrincipal`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `AttachThingPrincipal`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Attach a principal to an AWS IoT thing.
```

```
/*!
  \param principal: A principal to attach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
 client.AttachThingPrincipal(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir AttachThingPrincipalla section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour joindre un certificat à votre objet

L'`attach-thing-principal`exemple suivant attache un certificat à l' MyTemperatureSensor objet. Le certificat est identifié par un ARN. Vous trouverez l'ARN d'un certificat dans la console AWS IoT.

```
aws iot attach-thing-principal \
    --thing-name MyTemperatureSensor \
    --principal arn:aws:iot:us-
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez la section Comment gérer les objets avec le registre dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous AttachThingPrincipalà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to
an IoT Thing.
    * If the request is successful, it prints a confirmation message and
additional information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn)
{
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
```

```
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing
successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing.
HTTP Status Code: " +
                            attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, voir AttachThingPrincipalla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous AttachThingPrincipalà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`CreateKeysAndCertificate`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `CreateKeysAndCertificate`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if
 provided.
/*!
  \param outputFolder: Location for storing output in files, ignored when string
 is empty.
  \param certificateARNResult: A string to receive the ARN of the created
 certificate.
  \param certificateID: A string to receive the ID of the created certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
 createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
            client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ",
 certificate ID: "
                  << certificateID << std::endl;

        if (!outputFolder.empty()) {
```

```
            std::cout << "Writing certificate and keys to the folder '" <<
 outputFolder
                      << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/
certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
 certificateFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            certificateFile << outcome.GetResult().GetCertificatePem();
            certificateFile.close();

            const Aws::IoT::Model::KeyPair &keyPair =
 outcome.GetResult().GetKeyPair();

            Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
            std::ofstream privateKeyFile(privateKeyFilePath);
            if (!privateKeyFile.is_open()) {
                std::cerr << "Error opening private key file, '" <<
 privateKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            privateKeyFile << keyPair.GetPrivateKey();
            privateKeyFile.close();

            Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
            std::ofstream publicKeyFile(publicKeyFilePath);
            if (!publicKeyFile.is_open()) {
                std::cerr << "Error opening public key file, '" <<
 publicKeyFilePath
                          << "'."
                          << std::endl;
                return false;
            }
            publicKeyFile << keyPair.GetPublicKey();
        }
```

```
    }
    else {
        std::cerr << "Error creating keys and certificate: "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir CreateKeysAndCertificatela section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour créer une paire de clés RSA et émettre un certificat X.509

Ce qui suit `create-keys-and-certificate` crée une paire de clés RSA 2048 bits et émet un certificat X.509 à l'aide de la clé publique émise. Comme c'est la seule fois où l' AWS IoT fournit la clé privée pour ce certificat, veillez à le conserver dans un endroit sécurisé.

```
aws iot create-keys-and-certificate \
    --certificate-pem-outfile "myTest.cert.pem" \
    --public-key-outfile "myTest.public.key" \
    --private-key-outfile "myTest.private.key"
```

Sortie :

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
```

```
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nMMEXAMPLEuuN/
dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y
+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB
\n-----END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omittted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

Pour plus d'informations, voir Création et enregistrement d'un certificat d'appareil AWS IoT dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous CreateKeysAndCertificateà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Creates an IoT certificate asynchronously.
```

```
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT
certificate.
     * If the request is successful, it prints the certificate details and
returns the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }
```

- Pour plus de détails sur l'API, voir CreateKeysAndCertificatela section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun createCertificate(): String? {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CreateKeysAndCertificateà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`CreateThing`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `CreateThing`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Create an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
            createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir CreateThingla section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Exemple 1 : pour créer un enregistrement d'objet dans le registre

L'`create-thing`exemple suivant crée une entrée pour un appareil dans le registre AWS des objets IoT.

```
aws iot create-thing \
    --thing-name SampleIoTThing
```

Sortie :

```
{
    "thingName": "SampleIoTThing",
    "thingArn": "arn:aws:iot:us-west-2: 123456789012:thing/SampleIoTThing",
    "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "
}
```

Exemple 2 : pour définir un objet associé à un type d'objet

L'`create-thing`exemple suivant crée un objet doté du type d'objet spécifié et de ses attributs.

```
aws iot create-thing \
    --thing-name "MyLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Sortie :

```
{
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797"
}
```

Pour plus d'informations, consultez la section Comment gérer les objets avec le registre et les types d'objets dans le Guide du développeur de l'AWS IoT.

- Pour plus de détails sur l'API, reportez-vous CreateThingà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /**
    * Creates an IoT Thing with the specified name asynchronously.
    *
    * @param thingName The name of the IoT Thing to create.
    *
    * This method initiates an asynchronous request to create an IoT Thing with
the specified name.
    * If the request is successful, it prints the name of the thing and its ARN
value.
    * If an exception occurs, it prints the error message.
    */
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The
ARN value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
```

```
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, voir CreateThingla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CreateThingà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`CreateTopicRule`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `CreateTopicRule`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
  \param ruleName: The name for the rule.
  \param snsTopic: The SNS topic ARN for the action.
  \param sql: The SQL statement used to query the topic.
  \param roleARN: The IAM role ARN for the action.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
 &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);
```

```
    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
 std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir CreateTopicRulela section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour créer une règle qui envoie une alerte Amazon SNS

L'`create-topic-rule`exemple suivant crée une règle qui envoie un message Amazon SNS lorsque les relevés du niveau d'humidité du sol, tels qu'ils apparaissent dans l'ombre d'un appareil, sont faibles.

```
aws iot create-topic-rule \
    --rule-name "LowMoistureRule" \
    --topic-rule-payload file://plant-rule.json
```

L'exemple nécessite que le code JSON suivant soit enregistré dans un fichier nommé `plant-rule.json` :

```
{
    "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE
 state.reported.moisture = 'low'\n",
    "description": "Sends an alert whenever soil moisture level readings are too
 low.",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
            "sns": {
                "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyRPiLowMoistureTopic",
                "roleArn": "arn:aws:iam::123456789012:role/service-role/
MyRPiLowMoistureTopicRole",
                "messageFormat": "RAW"
            }
    }]
}
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez la section Création d'une règle AWS IoT dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous CreateTopicRuleà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
 actions.
```

```
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
```

```
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, voir CreateTopicRulela section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }
```

```
    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous CreateTopicRuleà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`DeleteCertificate`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DeleteCertificate`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Delete a certificate.
/*!
  \param certificateID: The ID of a certificate.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
 iotClient.DeleteCertificate(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir DeleteCertificatela section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour supprimer un certificat d'appareil

L'`delete-certificate`exemple suivant supprime le certificat de l'appareil avec l'ID spécifié.

```
aws iot delete-certificate \
    --certificate-
id c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbee1428d216d54d53ac9
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez DeleteCertificatele manuel de référence des API AWS IoT.

- Pour plus de détails sur l'API, reportez-vous DeleteCertificateà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
```

```
                System.out.println(certificateArn + " was successfully
 deleted.");
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, voir DeleteCertificatela section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteCertificateà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`DeleteThing`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DeleteThing`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Delete an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir DeleteThingla section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour afficher des informations détaillées sur un objet

L'`delete-thing`exemple suivant supprime un objet du registre AWS IoT de votre AWS compte.

aidez-moi à supprime-thing --thing-name « » FourthBulb

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez la section Comment gérer les objets avec le registre dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous DeleteThingà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
```

```
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, voir DeleteThingla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DeleteThingà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`DeleteTopicRule`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DeleteTopicRule`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Delete an AWS IoT rule.
/*!
  \param ruleName: The name for the rule.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
            request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
                  ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir DeleteTopicRulela section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour supprimer une règle

L'`delete-topic-rule`exemple suivant supprime la règle spécifiée.

```
aws iot delete-topic-rule \
    --rule-name "LowMoistureRule"
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez la section Supprimer une règle dans le Guide du développeur de l'AWS IoT.

- Pour plus de détails sur l'API, reportez-vous DeleteTopicRuleà la section Référence des AWS CLI commandes.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`DescribeEndpoint`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DescribeEndpoint`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Describe the endpoint specific to the AWS account making the call.
/*!
  \param endpointResult: String to receive the endpoint result.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
            "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
 iotClient.DescribeEndpoint(
            describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
```

```
        std::cerr << "Error describing endpoint" <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir DescribeEndpointla section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Exemple 1 : pour obtenir votre point de AWS terminaison actuel

L'`describe-endpoint`exemple suivant récupère le point de AWS terminaison par défaut auquel toutes les commandes sont appliquées.

```
aws iot describe-endpoint
```

Sortie :

```
{
    "endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

Pour plus d'informations, consultez DescribeEndpointle Guide du développeur de AWS l'IoT.

Exemple 2 : pour obtenir votre point de terminaison ATS

L'`describe-endpoint`exemple suivant récupère le point de terminaison Amazon Trust Services (ATS).

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

Sortie :

```
{
    "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

Pour plus d'informations, consultez la section Certificats X.509 et AWS IoT dans le Guide du développeur de l'AWS IoT.

- Pour plus de détails sur l'API, reportez-vous DescribeEndpointà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of
 the IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Da
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
```

```
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- Pour plus de détails sur l'API, voir DescribeEndpointla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
```

```
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DescribeEndpointà la section AWS SDK pour la référence de l'API Kotlin.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error>
 {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- Pour plus de détails sur l'API, voir DescribeEndpointla section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`DescribeThing`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DescribeThing`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
 iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "'" <<
 std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
                  << std::endl;
```

```
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
 attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir DescribeThingla section Référence des AWS SDK pour C ++ API.

CLI

AWS CLI

Pour afficher des informations détaillées sur un objet

L'`describe-thing`exemple suivant affiche des informations sur un objet (appareil) défini dans le registre AWS IoT de votre AWS compte.

est-ce que je ne décrivais pas --thing-name « » MyLightBulb

Sortie :

```
{
    "defaultClientId": "MyLightBulb",
    "thingName": "MyLightBulb",
    "thingId": "40da2e73-c6af-406e-b415-15acae538797",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    },
    "version": 1
```

```
}
```

Pour plus d'informations, consultez la section Comment gérer les objets avec le registre dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous DescribeThingà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " +
describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, voir DescribeThingla section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
```

```
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DescribeThingà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`DetachThingPrincipal`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DetachThingPrincipal`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Detach a principal from an AWS IoT thing.
/*!
  \param principal: A principal to detach.
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);
```

```
    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
 iotClient.DetachThingPrincipal(
           detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
 thing "
                  << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                  << thingName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, voir DetachThingPrincipalla section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour détacher une chose certificate/principal d'une chose

L'`detach-thing-principal`exemple suivant supprime un certificat représentant un mandant de l'objet spécifié.

```
aws iot detach-thing-principal \
    --thing-name "MyLightBulb" \
    --principal "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez la section Comment gérer les objets avec le registre dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous DetachThingPrincipalà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /**
    * Detaches a principal (certificate) from an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to detach.
    *
    * This method initiates an asynchronous request to detach a certificate from
an IoT Thing.
    * If the detachment is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void detachThingPrincipal(String thingName, String certificateArn) {
       DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
           .principal(certificateArn)
           .thingName(thingName)
           .build();

       CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully removed
from " + thingName);
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, reportez-vous DetachThingPrincipalà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous DetachThingPrincipalà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`ListCertificates`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `ListCertificates`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! List certificates registered in the AWS account making the call.
/*!
   \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listCertificates(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
 iotClient.ListCertificates(
                request);
```

```
        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
 outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                   result.GetCertificates().begin(),
                                   result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
 std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
 std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
                  << std::endl;
        std::cout << std::endl;
    }

    return true;
}
```

- Pour plus de détails sur l'API, reportez-vous ListCertificatesà la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Exemple 1 : Pour répertorier les certificats enregistrés dans votre AWS compte

L'`list-certificates`exemple suivant répertorie tous les certificats enregistrés dans votre compte. Si vous avez dépassé la limite de pagination par défaut de 25, vous pouvez utiliser la valeur de `nextMarker` réponse de cette commande et la fournir à la commande suivante

pour obtenir le prochain lot de résultats. Répétez jusqu'à ce qu'il `nextMarker` revienne sans valeur.

```
aws iot list-certificates
```

Sortie :

```
{
    "certificates": [
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36
            "certificateId":
 "604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
            "status": "ACTIVE",
            "creationDate": 1556810537.617
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e
            "certificateId":
 "262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
            "status": "ACTIVE",
            "creationDate": 1546447050.885
        },
        {
            "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "certificateId":
 "b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
            "status": "ACTIVE",
            "creationDate": 1546292258.322
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42
            "certificateId":
 "7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
            "status": "ACTIVE",
            "creationDate": 1541457693.453
        },
        {
```

```
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3
            "certificateId":
 "54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
            "status": "ACTIVE",
            "creationDate": 1541113568.611
        },
        {
            "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e
            "certificateId":
 "4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
            "status": "ACTIVE",
            "creationDate": 1541022751.983
        }
    ]
}
```

- Pour plus de détails sur l'API, reportez-vous ListCertificatesà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
 getAsyncClient().listCertificates();
```

```
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, reportez-vous ListCertificatesà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun listCertificates() {
    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
```

```
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous ListCertificatesà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`ListThings`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `ListThings`.

CLI

AWS CLI

Exemple 1 : pour répertorier tous les éléments du registre

L'`list-things`exemple suivant répertorie les objets (appareils) définis dans le registre AWS IoT de votre AWS compte.

```
aws iot list-things
```

Sortie :

```
{
    "things": [
        {
            "thingName": "ThirdBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
```

```
            },
            "version": 2
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        },
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
        "thingName": "SampleIoTThing",
        "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
        "attributes": {},
        "version": 1
        }
    ]
}
```

Exemple 2 : pour répertorier les éléments définis dotés d'un attribut spécifique

L'`list-things`exemple suivant affiche une liste d'objets dotés d'un attribut nomméwattage.

```
aws iot list-things \
    --attribute-name wattage
```

Sortie :

```
{
```

```
    "things": [
        {
            "thingName": "MyLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingTypeName": "LightBulb",
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3
        }
    ]
}
```

Pour plus d'informations, consultez la section Comment gérer les objets avec le registre dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous ListThingsà la section Référence des AWS CLI commandes.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
async fn show_things(client: &Client) -> Result<(), Error> {
```

```
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name:  {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type:  {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN:   {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }

    println!();

    Ok(())
}
```

- Pour plus de détails sur l'API, voir ListThingsla section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`SearchIndex`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `SearchIndex`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html
/*!
  \param: query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
 iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
 outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
```

```
            nextToken = result.GetNextToken();

        }
        else {
            std::cerr << "Error in SearchIndex: " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
 std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
                  << std::endl;
    }
    return true;
}
```

- Pour plus de détails sur l'API, reportez-vous SearchIndexà la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour interroger l'index des objets

L'`search-index`exemple suivant interroge l'`AWS_Things`index pour les objets de type`LightBulb`.

```
aws iot search-index \
    --index-name "AWS_Things" \
    --query-string "thingTypeName:LightBulb"
```

Sortie :

```
{
    "things": [
        {
```

```
            "thingName": "MyLightBulb",
            "thingId": "40da2e73-c6af-406e-b415-15acae538797",
            "thingTypeName": "LightBulb",
            "thingGroupNames": [
                "LightBulbs",
                "DeadBulbs"
            ],
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "ThirdBulb",
            "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        },
        {
            "thingName": "MyOtherLightBulb",
            "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "connectivity": {
                "connected": false
            }
        }
    ]
}
```

Pour plus d'informations, consultez la section Managing Thing Indexing dans le Guide du développeur de l'AWS IoT.

- Pour plus de détails sur l'API, reportez-vous SearchIndexà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
```

```
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
 cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, reportez-vous SearchIndexà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
```

```
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
 ${thing.thingId}") }
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous SearchIndexà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`UpdateIndexingConfiguration`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `UpdateIndexingConfiguration`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Update the indexing configuration.
/*!
  \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
  \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration
 object which is ignored if not set.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(
```

```
        const Aws::IoT::Model::ThingIndexingConfiguration
 &thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
 &thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {

 request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
 iotClient.UpdateIndexingConfiguration(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
                  << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous UpdateIndexingConfigurationà la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour activer l'indexation des objets

L'`update-indexing-configuration`exemple suivant active l'indexation des objets pour prendre en charge la recherche dans les données de registre, les données fictives et l'état de connectivité des objets à l'aide de l'index AWS_Things.

```
aws iot update-indexing-configuration
    --thing-indexing-
configuration thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

Cette commande ne produit aucun résultat.

Pour plus d'informations, consultez la section Managing Thing Indexing dans le Guide du développeur AWS IoT.

- Pour plus de détails sur l'API, reportez-vous UpdateIndexingConfigurationà la section Référence des AWS CLI commandes.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

Utilisation **`UpdateThing`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `UpdateThing`.

C++

Kit de développement logiciel (SDK) for C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
//! Update an AWS IoT thing with attributes.
/*!
  \param thingName: The name for the thing.
  \param attributeMap: A map of key/value attributes/
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
```

```
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
 &attributeMap,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous UpdateThingà la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour associer un objet à un type d'objet

L'`update-thing`exemple suivant associe un objet du registre AWS IoT à un type d'objet. Lorsque vous établissez l'association, vous fournissez des valeurs pour les attributs définis par le type d'objet.

```
aws iot update-thing \
```

```
    --thing-name "MyOtherLightBulb" \
    --thing-type-name "LightBulb" \
    --attribute-payload "{"attributes": {"wattage":"75", "model":"123"}}"
```

Cette commande ne produit pas de sortie. Utilisez la `describe-thing` commande pour voir le résultat.

Pour plus d'informations, consultez la section Types d'objets dans le Guide du développeur de l'AWS IoT.

- Pour plus de détails sur l'API, reportez-vous UpdateThingà la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /**
    * Updates the shadow of an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to update the shadow of an
IoT Thing.
    * If the request is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void updateShadowThing(String thingName) {
       // Create Thing Shadow State Document.
       String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
       SdkBytes data = SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8);
       UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
```

```
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " +
cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- Pour plus de détails sur l'API, reportez-vous UpdateThingà la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun updateThing(thingNameVal: String?) {
```

```
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient.fromEnvironment { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- Pour plus de détails sur l'API, reportez-vous UpdateThingà la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation AWS IoT avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes de SDK.

AWS IoTQuotas

Vous pouvez trouver des informations surAWS IoTQuotas dansAWSRéférence générale.

- PourAWS IoT Coreinformations sur les quotas, voirAWS IoT CorePoints de terminaison et quotas.
- PourAWS IoT Device Managementinformations sur les quotas, voirAWS IoT Device ManagementPoints de terminaison et quotas.
- PourAWS IoT Device Defenderinformations sur les quotas, voirAWS IoT Device DefenderPoints de terminaison et quotas.

Tarification d'AWS IoT Core

Vous trouverez des informations sur la AWS IoT Core tarification sur la page AWS Marketing et dans le calculateur de AWS prix.

- Pour vérifier les informations AWS IoT Core de tarification, consultez la section AWS IoT CoreTarification.
- Pour estimer le coût de votre solution d'architecte, consultez le calculateur de AWS prix.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.