



デベロッパーガイド

AWS Key Management Service



AWS Key Management Service: デベロッパーガイド

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

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

Table of Contents

AWS Key Management Service	1
使用する理由 AWS KMS	1
AWS KMS in AWS リージョン	2
AWS KMS 料金	2
AWS KMS サービスレベルアグリーメント	2
アクセス AWS Key Management Service	3
AWS Management Console	3
AWS KMS コンソールを使用するために必要なアクセス許可	3
AWS Command Line Interface	4
AWS KMS REST API	4
AWS SDKs	4
AWS SDKs の使用	5
AWS Encryption SDK	6
AWS KMS 結果整合性	6
ハイブリッドポスト量子 TLS	7
ポスト量子 TLS について	8
使用方法	9
ハイブリッドポスト量子 TLS を設定する	10
詳細はこちら	12
VPC エンドポイント AWS KMS を介して に接続する	12
の VPC エンドポイントを作成する AWS KMS	13
VPC エンドポイントへの接続	14
VPC エンドポイントを使用して AWS KMS リソースへのアクセスを制御する	15
VPC エンドポイントを使用する AWS KMS リクエストのログ記録	19
デュアルスタックのエンドポイント	20
IPv6 で使用できない機能	21
概念	22
序章	22
設計目標	23
AWS KMS keys	24
カスターマネージドキー	27
AWS マネージドキー	28
AWS 所有のキー	29
AWS KMS key 階層	29

キー識別子 (KeyId)	31
非対称キー	34
HMAC キー	36
ML-DSA キー	38
マルチリージョンキー	39
インポートされたキーマテリアル	50
CloudHSM キーストアの KMS キー	56
外部キーストアの KMS キー	59
AWS KMS 暗号化の必須事項	62
エントロピーと乱数生成	62
対称キーのオペレーション (暗号化のみ)	62
非対称キーのオペレーション (暗号化、デジタル署名、署名の検証)	63
キー導出関数	64
AWS KMS デジタル署名の内部使用	64
エンベロープ暗号化	64
暗号化オペレーション	66
サポートされているアルゴリズム	68
KMS キーのアクセスとアクセス許可	72
KMS キーポリシー	72
KMS キー向けグラント	73
キーポリシー	73
キーポリシーを作成する	74
デフォルトのキーポリシー	81
キーポリシーを表示する	97
キーポリシーを変更する	100
AWS サービスのアクセス許可	103
IAM ポリシー	104
複数の IAM プリンシパルが KMS キーにアクセスできるようにする	106
IAM ポリシーのベストプラクティス	106
IAM ポリシーステートメントで KMS キーを指定する	109
例	112
リソースコントロールポリシー	119
権限	122
グラントの概念	123
ベストプラクティス	128
グラントへのアクセスを制御する	129

グラントの作成	131
グラントの表示	139
グラントトークンを使用する	140
グラントの使用停止と取り消し	141
条件キー	142
AWS グローバル条件キー	143
AWS KMS 条件キー	147
AWS KMSAWS Nitro Enclaves の条件キー	215
最小特権のアクセス許可	219
最小特権のアクセス許可の実装	220
属性ベースのアクセスコントロール (ABAC)	223
の ABAC 条件キー AWS KMS	225
タグまたはエイリアス	227
の ABAC のトラブルシューティング AWS KMS	229
ロールベースのアクセスコントロール (RBAC)	233
クロスアカウントアクセス	235
ステップ 1: ローカルアカウントにキーポリシーステートメントを追加する	238
ステップ 2: 外部アカウントに IAM ポリシーを追加する	241
外部 KMS キーの使用を許可する AWS のサービス	243
他のアカウントで KMS キーを使用する	243
マルチリージョンキーへのアクセスを制御する	244
マルチリージョンキーの認可の原則	245
マルチリージョンキー管理者およびユーザーを認可する	246
アクセスの確認	251
キーポリシーを確認する	251
IAM ポリシーの確認	255
許可の確認	257
暗号化コンテキスト	258
暗号化コンテキストのルール	259
ポリシーでの暗号化コンテキスト	260
グラントでの暗号化コンテキスト	260
暗号化コンテキストのログ記録	261
暗号化コンテキストの保存	261
アクセス許可をテストする	262
DryRun とは	262
API で DryRun を指定する	263

アクセス AWS KMS 許可のトラブルシューティング	264
例 1: ユーザーが の KMS キーへのアクセスを拒否される AWS アカウント	266
例 2: ユーザーが別の で KMS キーを使用するアクセス許可を持つロールを引き受ける AWS アカウント	268
用語集	272
認証	272
Authorization	272
アイデンティティを使用した認証	272
ポリシーを使用したアクセスの管理	276
AWS KMS リソース	278
KMS キーを作成する	280
KMS キーを作成するためのアクセス許可	282
作成する KMS キーのタイプの選択	283
対称暗号化 KMS キーを作成する	285
非対称 KMS キーを作成する	291
HMAC KMS キーを作成する	297
マルチリージョンのプライマリキーを作成する	302
マルチリージョンのレプリカキーを作成する	307
ステップ 1: レプリカリージョンを選択する	308
ステップ 2: レプリカキーを作成する	309
インポートされたキーマテリアルを持つ KMS キーの作成	315
キーマテリアルをインポートするためのアクセス許可	316
インポートされたキーマテリアルの要件	317
ステップ 1: キーマテリアルなしで AWS KMS key を作成する	320
ステップ 2: ラップパブリックキーおよびインポートトークンのダウンロード	323
ステップ 3: キーマテリアルを暗号化する	332
ステップ 4: キーマテリアルのインポート	341
キーストアに KMS AWS CloudHSM キーを作成する	347
CloudHSM キーストアに新しい KMS キーを作成する	348
外部キーストアで KMS キーを作成する	354
外部キーストアの KMS キーの要件	356
外部キーストアで新しい KMS キーを作成する	357
キーの特定と表示	365
キー ID とキー ARN を検索する	365
KMS キーの詳細にアクセスして一覧表示する	367
さまざまなキータイプの特定	377

非対称 KMS キーを特定する	377
HMAC KMS キーの特定	378
マルチリージョン KMS キーの特定	379
インポートされたキーマテリアルを持つ KMS キーを特定する	380
キーストア内の KMS AWS CloudHSM キーを特定する	380
外部キーストアの KMS キーを特定する	381
コンソールの表示をカスタマイズする	382
KMS キーのソートおよびフィルタリング	383
KMS キーテーブルのカスタマイズ	385
キーストアで KMS キーと AWS CloudHSM キーマテリアルを検索する	387
キーストアで KMS AWS CloudHSM キーを検索する	388
キーストアのすべての AWS CloudHSM キーを検索する	389
キーの KMS AWS CloudHSM キーを検索する	391
KMS AWS CloudHSM キーのキーを検索する	396
キーの有効化と無効化	399
キーをローテーションする	402
KMS キーをローテーションする理由	404
キーローテーションの仕組み	405
自動キーローテーションを有効にする	409
自動キーローテーションを無効にする	412
オンデマンドキーローテーションの実行	413
オンデマンドキーローテーションの開始 (コンソール)	414
オンデマンドキーローテーションの開始 (AWS KMS API)	415
ローテーションとキーマテリアルを一覧表示する	417
ローテーションとキーマテリアルを一覧表示する (コンソール)	417
ローテーションとキーマテリアルを一覧表示する (AWS KMS API)	418
キーを手動でローテーションする	419
マルチリージョンキーセットのプライマリキーを変更する	422
プライマリリージョンを更新する	424
キーの削除	427
待機期間について	428
特別な考慮事項	428
キー削除へのアクセスを制御する	431
キー管理者に、キー削除のスケジュールおよびキャンセルを許可する	432
キー削除をスケジュールする	434
.....	435

キー削除をキャンセルする	436
アラームの作成	437
KMS キーの過去の使用状況を確認する	439
KMS キーのアクセス許可内容を確認し、潜在的な使用の範囲を判断する	440
AWS CloudTrail ログを調べて実際の使用状況来判断する	440
インポートされたキーマテリアルを削除する	443
データキーを生成する	446
データキーの作成	446
データキーを使用した暗号化オペレーションの仕組み	447
データキーでデータを暗号化する	447
データキーでデータを復号する	448
使用できない KMS キーがデータキーに及ぼす影響	449
データキーペアを生成する	451
データキーペアを作成する	451
データキーペアを使用した暗号化オペレーションの仕組み	452
データキーペアでデータを暗号化する	453
データキーペアでデータを復号する	453
データキーペアでメッセージに署名する	454
データキーペアを使用した署名の検証	455
データキーペアを用いて共有シークレットを取得する	456
パブリックキーを使用してオフラインオペレーションを実行する	457
パブリックキーのダウンロードに関する特別な考慮事項	458
パブリックキーをダウンロードする	459
オフラインオペレーションの例	461
オフラインで共有シークレットを取得する	461
ML-DSA キーペアによるオフライン検証	462
SM2 キーペアによるオフライン検証 (中国リージョンのみ)	465
キーをモニタリングする	471
モニタリングツール	472
自動ツール	472
手動ツール	473
を使用したログ記録 AWS CloudTrail	474
CloudTrail での AWS KMS ログエントリの検索	475
証跡からの AWS KMS イベントの除外	476
AWS KMS ログエントリの例	477
CloudWatch でキーをモニタリングする	561

AWS KMS メトリクスとディメンション	561
インポートされたキーマテリアルの有効期限を通知する CloudWatch アラームを作成する	571
外部キーストアの CloudWatch アラームを作成する	572
Amazon EventBridge を使用したキーのモニタリング	576
KMS CMK ローテーション	577
KMS でインポートされたキーマテリアルの有効期限	577
KMS CMK 削除	578
エイリアス	580
エイリアスの仕組み	581
エイリアスへのアクセスの制御	584
kms:CreateAlias	584
kms:ListAliases	586
kms:UpdateAlias	586
kms>DeleteAlias	588
エイリアスのアクセス許可を制限する	589
エイリアスを作成する	590
エイリアス名とエイリアス ARN を見つける	593
エイリアスを更新する	598
エイリアスを削除する	599
エイリアスを使用して KMS キーへのアクセスを制御する	600
kms:RequestAlias	602
kms:ResourceAliases	603
アプリケーションでエイリアスを使用する方法について説明します。	604
AWS CloudTrail ログでエイリアスを検索する	606
[タグ]	608
タグへのアクセスを制御する	609
ポリシーのタグ付け許可	610
タグ付け許可を制限する	612
タグを追加する	614
KMS キーの作成中にタグを追加する	614
既存の KMS キーにタグを追加する	615
タグの編集	617
タグを削除する	618
タグの表示	619
タグを使用して KMS キーへのアクセスを制御する	621

キーストア	625
AWS KMS 標準キーストア	625
AWS KMS インポートされたキーマテリアルを持つ標準キーストア	625
AWS KMS カスタムキーストア	627
AWS CloudHSM キーストア	627
外部キーストア	628
AWS CloudHSM キーストア	628
AWS CloudHSM キーストア の概念	631
AWS CloudHSM キーストア へのアクセスを制御する	634
AWS CloudHSM キーストア を作成する	636
AWS CloudHSM キーストア を表示する	643
AWS CloudHSM キーストア 設定を編集する	646
AWS CloudHSM キーストア を接続する	650
AWS CloudHSM キーストア を切断する	654
AWS CloudHSM キーストア を削除する	657
カスタムキーストアのトラブルシューティング	659
外部キーストア	675
外部キーストアのコンセプト	679
外部キーストアの仕組み	688
外部キーストア へのアクセスの制御	690
プロキシ接続オプションを選択する	695
外部キーストア を作成する	707
外部キーストア プロパティを編集する	720
外部キーストア を表示する	726
外部キーストア をモニタリングする	732
外部キーストア を接続および切断する	744
外部キーストア を削除する	754
外部キーストアのトラブルシューティング	756
セキュリティ	781
データ保護	782
キーマテリアルの保護	782
データ暗号化	783
インターネットのプライバシー	785
Identity and Access Management	786
AWS マネージドポリシー	786
サービスにリンクされた役割	790

ログ記録とモニタリング	796
コンプライアンス検証	798
コンプライアンスとセキュリティに関するドキュメント	798
詳細	799
耐障害性	799
リージョンの隔離	800
マルチテナント設計	800
の耐障害性のベストプラクティス AWS KMS	801
インフラストラクチャセキュリティ	801
物理ホストの分離	803
クォータ	804
リソースクォータ	804
AWS KMS keys: 100,000	805
KMS キーごとのエイリアス: 50	805
KMS キーあたりのグラント: 50,000	806
カスタムキーストアのリソースクォータ: 10	806
オンデマンドローテーション: 10	806
クォータのリクエスト	807
各 AWS KMS API オペレーションのリクエストクォータ	808
リクエストクォータの適用	815
暗号化オペレーションの共有クォータ	815
ユーザーに代わって API が実行するリクエスト	817
クロスアカウントリクエスト	817
カスタムキーストアのリクエストクォータ	817
リクエストのスロットリング	819
コードの例	821
基本	825
こんにちは AWS KMSは	826
基本を学ぶ	829
アクション	903
シナリオ	1057
テーブルの暗号化を操作する	1057
AWS Nitro Enclaves の暗号化アテストレーション	1060
Nitro Enclave AWS KMS APIs を呼び出す方法	1061
Nitro Enclaves に対するリクエストの監視	1062
Decrypt (エンクレープ用)	1063

GenerateDataKey (エンクレーブ用)	1064
GenerateDataKeyPair (エンクレーブ用)	1065
GenerateRandom (エンクレーブ用)	1066
AWS サービスの暗号化	1068
Amazon Elastic Block Store (Amazon EBS)	1069
Amazon EBS 暗号化	1069
KMS キーとデータキーを使用する	1070
Amazon EBS 暗号化コンテキスト	1070
Amazon EBS 障害の検出	1071
AWS CloudFormation を使用して暗号化された Amazon EBS ボリュームを作成する	1072
Amazon EMR	1072
EMR ファイルシステム (EMRFS) のデータを暗号化する	1073
クラスターノードのストレージボリュームのデータを暗号化する	1075
暗号化コンテキスト	1076
Amazon Redshift	1077
Amazon Redshift 暗号化	1078
暗号化コンテキスト	1079
参照資料	1080
キーステータスリファレンス	1081
キーステータスと KMS キーの種類	1081
キーステータスの表	1082
キータイプリファレンス	1090
キータイプの表	1091
特殊な機能の表	1097
キー仕様のリファレンス	1106
SYMMETRIC_DEFAULT キー仕様	1108
RSA キー仕様	1109
楕円曲線のキー仕様	1113
HMAC KMS キーの主な仕様	1115
ML-DSA キー仕様	1115
SM2 キー仕様 (中国リージョンのみ)	1116
アクセス許可に関するリファレンス	1117
列の説明	1162
AWS KMS 内部オペレーション	1164
ドメインとドメインの状態	1165
内部通信セキュリティ	1169

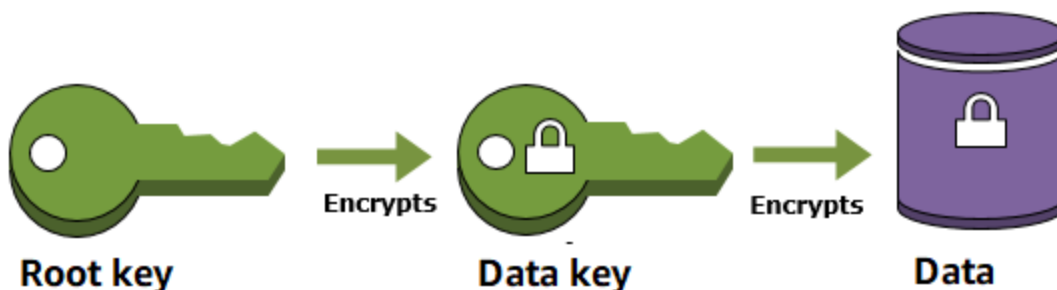
マルチリージョンキーのレプリケーションプロセス	1172
耐久性の保護	1173
ドキュメント履歴	1174
最新の更新	1174
以前の更新	1180
.....	mclxxxv

AWS Key Management Service

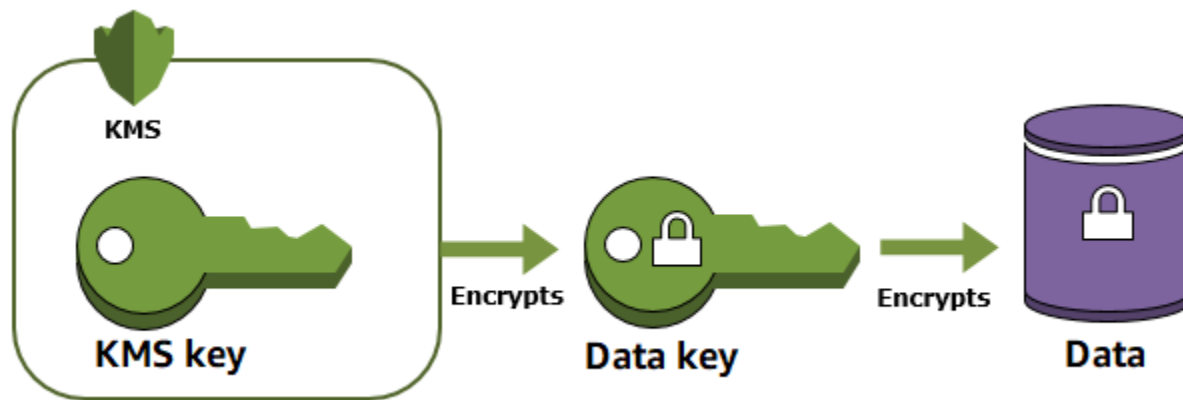
AWS Key Management Service (AWS KMS) は、データの暗号化に使用される暗号化キーの作成と制御を容易にする AWS マネージドサービスです。で AWS KMS keys 作成した AWS KMS は、[FIPS 140-3 セキュリティレベル 3 検証済みハードウェアセキュリティモジュール \(HSM\)](#) によって保護されています。暗号化 AWS KMS されていないままになることはありません。KMS キーを使用または管理するには、 を操作します AWS KMS。

を使用する理由 AWS KMS

データを暗号化する場合は、暗号化キーを保護する必要があります。キーを暗号化する場合は、その暗号化キーを保護する必要があります。最終的には、データを保護する階層の最高レベルの暗号化キー (ルートキーと呼ばれる) を保護する必要があります。そこで AWS KMS が登場します。



AWS KMS はルートキーを保護します。KMS キーは、内で完全に作成、管理、使用、削除されます AWS KMS。サービスが暗号化されていないままの状態に置かれることはありません。KMS キーを使用または管理するには、 を呼び出します AWS KMS。



さらに、[でキーポリシー](#)を作成および管理して AWS KMS、信頼できるユーザーのみが KMS キーにアクセスできるようにすることができます。

AWS KMS in AWS リージョン

AWS リージョン AWS KMS がサポートされている は、[AWS Key Management Service 「エンドポイントとクォータ」](#)に記載されています。がサポート AWS リージョン する で AWS KMS 機能 AWS KMS がサポートされていない場合、その機能に関するトピックでリージョンの違いについて説明します。

AWS KMS 料金

他の AWS 製品と同様に、AWS KMS を使用する場合、契約や最低購入額は必要ありません。AWS KMS 料金の詳細については、[AWS Key Management Service 「料金表」](#)を参照してください。

AWS KMS サービスレベルアグリーメント

AWS Key Management Service は、当社の[サービス可用性ポリシーを定義するサービスレベルアグリーメント](#)に基づいています。

アクセス AWS Key Management Service

は以下の AWS KMS 方法で を操作できます。

トピック

- [AWS Management Console](#)
- [AWS Command Line Interface](#)
- [AWS KMS REST API](#)
- [AWS SDKs](#)
- [AWS SDK でのこのサービスの使用](#)
- [AWS Encryption SDK](#)
- [AWS KMS 結果整合性](#)
- [でのハイブリッドポスト量子 TLS の使用 AWS KMS](#)
- [VPC エンドポイント AWS KMS を介して に接続する](#)
- [デュアルスタックエンドポイントのサポート](#)

AWS Management Console

コンソールは、AWS KMS および AWS リソースを管理するためのウェブベースのユーザーインターフェイスです。にサインアップしている場合は AWS アカウント、 にサインイン AWS Management Console し、AWS Management Console ホームページ AWS KMS から を選択して、AWS KMS コンソールにアクセスできます。

AWS KMS コンソールを使用するために必要なアクセス許可

AWS KMS コンソールを使用するには、ユーザーは 内の AWS KMS リソースを操作できる最小限のアクセス許可のセットを持っている必要があります AWS アカウント。これらの AWS KMS アクセス許可に加えて、ユーザーには、IAM ユーザーおよび IAM ロールを一覧表示するためのアクセス許可も必要です。これらの最小限必要なアクセス許可よりも制限された IAM ポリシーを作成した場合、AWS KMS コンソールは、その IAM ポリシーを使用するユーザーの意図したとおりには機能しません。

AWS KMS コンソールへの読み取り専用アクセスをユーザーに許可するために必要な最小限のアクセス権限については、「[AWS KMS コンソールで KMS キーの表示をユーザーに許可する](#)」を参照してください。

ユーザーが AWS KMS コンソールを使用して KMS キーを作成および管理できるようにするには、「」の説明に従って、AWSKeyManagementServicePowerUser 管理ポリシーをユーザーにアタッチします[AWS の 管理ポリシー AWS Key Management Service](#)。

[AWS SDK](#)、[AWS Command Line Interface](#)、もしくは [AWS Tools for PowerShell](#) を経由して AWS KMS API を使用するユーザーに対して、最小限のコンソールアクセス許可を付与する必要はありません。ただし、API を使用する権限をこれらのユーザーに付与する必要があります。詳細については、「[アクセス許可に関するリファレンス](#)」を参照してください。

AWS Command Line Interface

AWS CLI ツールを使用して、システムのコマンドラインでコマンドを発行したり、スクリプトを構築したりして、タスク AWS (含む AWS KMS) を実行できます。

AWS KMS で を使用する方法の詳細については AWS CLI、[AWS CLI 「 コマンドリファレンス」](#)を参照してください。

AWS KMS REST API

のアーキテクチャ AWS KMS は、プログラミング言語に依存しないように設計されています。REST API は への HTTP インターフェイスです AWS KMS。ではREST API、標準の HTTP リクエストを使用してキーを作成、フェッチ、削除します。

の使用の詳細については AWS KMS REST API、[AWS Key Management Service API リファレンス](#)を参照してください。

AWS SDKs

AWS はSDKs (ソフトウェア開発キット) を提供します。JavaScript AWS SDKs を使用すると、AWS KMS および へのプログラムによるアクセスを簡単に作成 AWSできます。AWS KMS はREST サービスです。基盤となる AWS KMS REST API をラップし、プログラミングタスクを簡素化する AWS SDK ライブラリ AWS KMS を使用して、 にリクエストを送信できます。ダウンロードしてインストールする方法など AWS SDKs、の詳細については、「[で構築するツール AWS](#)」を参照してください。

[AWS SDKs AWS KMS を使用するためのコード例](#) は、AWS KMS を通じて を使用するための適切な出発点を提供します AWS SDKs。

AWS SDK でのこのサービスの使用

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

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

可用性の例

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

AWS Encryption SDK

AWS Encryption SDK は、アプリケーションにクライアント側の暗号化を実装するためのツールです。KMS へのフルアクセスは提供しませんが、代わりに と統合するか AWS KMS、KMS キーを参照せずにスタンドアロン SDK として使用できます。Java、JavaScript、C、Python、およびその他のプログラミング言語のライブラリをご用意しています。

詳細については、[AWS Encryption SDK デベロッパーガイド](#)を参照してください。

AWS KMS key ポリシーと IAM ポリシー

AWS KMS 結果整合性

AWS KMS API は、システムの分散性により、[結果整合性](#)モデルに従います。その結果、AWS KMS リソースへの変更は、実行した後続のコマンドにすぐに表示されない場合があります。

AWS KMS API コールを実行すると、変更が全体で利用可能になるまでに少し時間がかかる場合があります。通常、変更がシステム全体に反映されるまでに数秒もかかりませんが、場合によっては数分かかることがあります。この間、`InvalidStateException` または `NotFoundException` などの予期しないエラーが発生することがあります。たとえば、`GetParametersForImport` を呼び出した直後に `NotFoundException` と、`CreateKey` を返す AWS KMS 可能性があります。

短い待機期間後にオペレーションを自動的に再試行するように、AWS KMS クライアントに再試行戦略を設定することをお勧めします。詳細については、AWS SDK とツールのリファレンスガイドの「[再試行動作](#)」を参照してください。

API 呼び出し関連のグラントでは、[グラントトークンを使用](#)して潜在的な遅延を回避し、グラント内のアクセス許可をすぐに使用できます。詳細については、「[結果整合性 \(グラント用\)](#)」を参照してください。

でのハイブリッドポスト量子 TLS の使用 AWS KMS

AWS Key Management Service (AWS KMS) は、Transport Layer Security (TLS) ネットワーク暗号化プロトコルのハイブリッドポスト量子キー交換オプションをサポートしています。この TLS オプションは、AWS KMS API エンドポイントに接続するときに使用できます。これらのオプションのハイブリッドポスト量子キー交換機能は、現在使用している TLS 暗号化と同等以上に安全であり、セキュリティ上のさらなる長期的な利点をもたらす可能性があります。ただし、現在使用されている従来のキー交換プロトコルと比較して、レイテンシーとスループットに影響します。

AWS Key Management Service (AWS KMS) に送信するデータは、Transport Layer Security (TLS) 接続によって提供される暗号化によって転送中に保護されます。AWS KMS が TLS セッションでサポートしている従来の暗号スイートにより、現在のテクノロジーではキー交換メカニズムに対するブルートフォース攻撃は実行不可能です。しかし、大規模な量子コンピューティングが将来実用的になると、TLS 鍵交換メカニズムで使用される従来の暗号スイートは、これらの攻撃の影響を受けやすくなります。TLS 接続を介して渡されるデータの長期的な機密性に依存するアプリケーションを開発している場合は、大規模な量子コンピュータが使用できるようになる前に、ポスト量子暗号化に移行する計画を検討する必要があります。AWS は、この未来に備えており、準備も万端です。

現在暗号化されているデータを潜在的な将来の攻撃から保護するために、AWS は量子耐性アルゴリズムまたはポスト量子アルゴリズムの開発に暗号コミュニティに参加しています。クラシック要素とポスト量子要素を組み合わせたハイブリッドポスト量子キー交換暗号スイート AWS KMS をに実装し、TLS 接続がクラシック暗号スイートと同等以上に強力になるようにしました。

これらのハイブリッド暗号スイートは、[ほとんどの AWS リージョン](#)本稼働ワークロードで使用できます。ただし、ハイブリッド暗号スイートのパフォーマンス特性と帯域幅要件は従来のキー交換メカニズムとは異なるため、さまざまな条件下で [AWS KMS API コールでテスト](#)することをお勧めします。

フィードバック

これまでと同様、皆様のフィードバックや私たちのオープンソースリポジトリへの参加はいつでも歓迎です。特に、この新しい種類の TLS トラフィックが皆様のインフラストラクチャとどのように相互作用するかをぜひお聞かせください。

- このトピックに関するフィードバックを提供するには、このページの右下隅にある [フィードバック] リンクを使用してください。
- これらのハイブリッド暗号スイートは、GitHub の [s2n-tls](#) リポジトリのオープンソースで開発されています。暗号スイートのユーザビリティに関するフィードバックを提供したり、新しいテスト条件や結果を共有したりするには、s2n-tls リポジトリで[課題を作成](#)してください。

- [aws-kms-pq-tls-example](#) GitHub リポジトリ AWS KMS の でハイブリッドポスト量子 TLS を使用するためのコードサンプルを記述しています。ハイブリッド暗号スイートを使用する AWS KMS ように HTTP クライアントを設定する方法について質問したり、アイデアを共有したりするには、aws-kms-pq-tls-example リポジトリで[問題を作成します](#)。

サポートされている AWS リージョン

用のポスト量子 TLS AWS KMS は、 が AWS KMS サポート AWS リージョン するすべての で使用できます。

それぞれの AWS KMS エンドポイントのリストについては AWS リージョン、の[AWS Key Management Service 「エンドポイントとクォータ」](#)を参照してくださいAmazon Web Services 全般のリファレンス。FIPS エンドポイントの詳細については、「Amazon Web Services 全般のリファレンス」の「[FIPS エンドポイント](#)」を参照してください。

TLS におけるハイブリッドポスト量子キー交換について

AWS KMS は、ハイブリッドポスト量子キー交換暗号スイートをサポートしています。Linux システムで AWS SDK for Java 2.x および AWS Common Runtime を使用して、これらの暗号スイートを使用する HTTP クライアントを設定できます。次に、HTTP クライアントを使用して AWS KMS エンドポイントに接続するたびに、ハイブリッド暗号スイートが使用されます。

この HTTP クライアントは、TLS プロトコルのオープンソース実装である [s2n-tls](#) を使用します。s2n-tls が使用するハイブリッド暗号スイートは、直接データ暗号化ではなく、キー交換専用の実装されています。キー交換中、クライアントとサーバーは、転送中にデータの暗号化と復号に使用するキーを計算します。

がs2n-tls使用するアルゴリズムは、TLS で現在使用されている従来のキー交換アルゴリズムである [Elliptic Curve Diffie-Hellman](#) (ECDHML-KEM) と、米国国立標準技術研究所 (NIST) が[最初の標準ポスト量子キーアグリーメントアルゴリズムとして指定したパブリックキー暗号化およびキー確立アルゴリズムである \[Module-Lattice-Based Key-Encapsulation Mechanism\]\(#\) \(\)](#) を組み合わせたハイブリッドです。このメカニズムは、各アルゴリズムを独立して使用してキーを生成します。次に、2つのキーを暗号的に組み合わせます。s2n-tls では、プリファレンスリストで ECDH と ML-KEM を最初に配置するポスト量子 TLS を優先して、[HTTP クライアントを設定](#)できます。互換性を確保するために、従来のキー交換アルゴリズムがプリファレンスリストに含まれていますが、プリファレンスの順序では低くなっています。

でのハイブリッドポスト量子 TLS の使用 AWS KMS

への呼び出しには、ハイブリッドポスト量子 TLS を使用できます AWS KMS。HTTP クライアントのテスト環境を設定するときは、次の点に注意してください。

転送時の暗号化

s2n-tls のハイブリッド暗号スイートは、転送中の暗号化のみに使用されます。クライアントから AWS KMS エンドポイントに移動する間、データは保護されます。AWS KMS は、これらの暗号スイートを使用してデータを暗号化しません AWS KMS keys。

代わりに、AWS KMS が KMS キーでデータを暗号化する場合、256 ビットキーによる対称暗号化と、すでに量子耐性がある Galois Counter Mode (AES-GCM) アルゴリズムの Advanced Encryption Standard を使用します。理論上の将来、256 ビット AES-GCM キーで作成された暗号文に対する大規模な量子コンピューティング攻撃は、[キーの効果的なセキュリティを 128 ビットに低下させます](#)。このセキュリティレベルは、AWS KMS 暗号文に対するブルートフォース攻撃を実行不可能にするのに十分です。

サポートされているシステム

s2n-tls のハイブリッド暗号スイートの使用は、現在 Linux システムでのみサポートされています。加えて、これらの暗号スイートは、AWS SDK for Java 2.xなどの AWS 共通ランタイムをサポートする SDK でのみサポートされます。例については、[ハイブリッドポスト量子 TLS を設定する](#)を参照してください。

AWS KMS エンドポイント

ハイブリッド暗号スイートを使用する場合は、標準の AWS KMS endpoint を使用します。AWS KMS は、[FIPS 140-3 検証済みエンドポイント](#)のハイブリッドポスト量子 TLS をサポートしていません。

s2n-tls でポスト量子 TLS 接続を優先して HTTP クライアントを設定すると、ポスト量子暗号は暗号プリファレンスリストの先頭になります。ただし、互換性を保つため、プリファレンスリストには優先順で低い従来の非ハイブリッド暗号が含まれています。FIPS 140-3 検証済みエンドポイントでポスト量子 TLS AWS KMS を優先するように HTTP クライアントを設定すると、s2n-tls は従来の非ハイブリッドキー交換暗号をネゴシエートします。

それぞれの AWS KMS エンドポイントのリストについては AWS リージョン、の[AWS Key Management Service 「エンドポイントとクォータ」](#)を参照してくださいAmazon Web Services 全

般のリファレンス。FIPS エンドポイントの詳細については、「Amazon Web Services 全般のリファレンス」の「[FIPS エンドポイント](#)」を参照してください。

ハイブリッドポスト量子 TLS を設定する

この手順では、AWS 共通ランタイム HTTP クライアントの Maven 依存関係を追加します。次に、ポスト量子 TLS を優先する HTTP クライアントを設定します。次に、HTTP AWS KMS クライアントを使用する クライアントを作成します。

AWS KMSでのハイブリッドポスト量子 TLS の設定と使用の完全な実例については、[aws-kms-pq-tls-example](#) リポジトリを参照してください。

Note

プレビュー版として利用可能な AWS 共通ランタイム HTTP クライアントは、2023 年 2 月に一般公開されました。このリリースでは、`tlsCipherPreference` クラスと `tlsCipherPreference()` メソッドパラメータが、`postQuantumTlsEnabled()` メソッドパラメータに置き換えられました。プレビュー中にこの例を使用していた場合は、コードを更新する必要があります。

1. AWS 共通ランタイムクライアントを Maven の依存関係に追加します。利用可能な最新バージョンを使用することをお勧めします。

たとえば、このステートメント2.30.22は AWS 、共通ランタイムクライアントのバージョンを Maven の依存関係に追加します。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
  <version>2.30.22</version>
</dependency>
```

2. ハイブリッドポスト量子暗号スイートを有効にするには、プロジェクト AWS SDK for Java 2.x に を追加して初期化します。続いて、次の例に示すように、HTTP クライアントでハイブリッドポスト量子暗号スイートを有効にします。

このコードは、`postQuantumTlsEnabled()`メソッドパラメータを使用して、推奨されるハイブリッドポスト量子暗号スイートである ECDH と を優先する[AWS 一般的なランタイム](#)

[HTTP クライアント](#)を設定しますML-KEM。次に、設定された HTTP クライアントを使用して、AWS KMS 非同期クライアントのインスタンス を構築します[KmsAsyncClient](#)。このコードが完了すると、KmsAsyncClient インスタンス上のすべての [AWS KMS API](#) リクエストは、ハイブリッドポスト量子 TLS を使用します。

```
// Configure HTTP client
SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
    .postQuantumTlsEnabled(true)
    .build();

// Create the AWS KMS async client
KmsAsyncClient kmsAsync = KmsAsyncClient.builder()
    .httpClient(awsCrtHttpClient)
    .build();
```

3. ハイブリッドポスト量子 TLS を使用して AWS KMS 呼び出しをテストします。

設定された AWS KMS クライアントで AWS KMS API オペレーションを呼び出すと、ハイブリッドポスト量子 TLS を使用して呼び出しが AWS KMS エンドポイントに送信されます。設定をテストするには、 などの AWS KMS API を呼び出します[ListKeys](#)。

```
ListKeysReponse keys = kmsAsync.listKeys().get();
```

ハイブリッドポスト量子 TLS 設定をテストする

が呼び出すアプリケーションで、ハイブリッド暗号スイートを使用して次のテストを実行することを検討してください AWS KMS。

- 負荷テストとベンチマークを実行します。ハイブリッド暗号スイートの動作は、従来のキー交換アルゴリズムとは異なります。ハンドシェイク時間が長くなるように、接続のタイムアウトを調整する必要がある場合があります。AWS Lambda 関数内で実行している場合は、実行タイムアウト設定を延長します。
- 別の場所からの接続を試します。要求が通過するネットワークパスによっては、ディープパケットインスペクション (DPI) を持つ中間ホスト、プロキシ、またはファイアウォールが要求をブロックしていることが検出されることがあります。これは、TLS ハンドシェイクの [ClientHello](#) 部分で新しい暗号スイートを使用した場合や、大きなキー交換メッセージから発生することがあります。これらの問題を解決できない場合は、セキュリティチームまたは IT 管理者と協力して、関連する構成を更新し、新しい TLS 暗号スイートのブロックを解除してください。

ポスト量子 TLS の詳細については、「」を参照してください。AWS KMS

でのハイブリッドポスト量子 TLS の使用の詳細については AWS KMS、以下のリソースを参照してください。

- ブログ投稿や研究論文へのリンクなど AWS、での量子後暗号化の詳細については、「[Post-Quantum Cryptography](#)」を参照してください。
- s2n-tls の詳細については、「[新しいオープンソース TLS 実装である s2n-tls の導入](#)」と「[s2n-tls の使用](#)」を参照してください。
- AWS 共通ランタイム HTTP クライアントの詳細については、「AWS SDK for Java 2.x デベロッパーガイド」の「[CRT AWS ベースの HTTP クライアントの設定](#)」を参照してください。
- 米国国立標準技術研究所 (NIST) のポスト量子暗号プロジェクトの詳細については、「[Post-Quantum Cryptography](#)」(ポスト量子暗号化) を参照してください。
- NIST ポスト量子暗号標準化については、「[Post-Quantum Cryptography Standardization](#)」(ポスト量子暗号標準化) を参照してください。

VPC エンドポイント AWS KMS を介して に接続する

Virtual Private Cloud (VPC) のプライベートインターフェイスエンドポイント AWS KMS を介して に直接接続できます。インターフェイス VPC エンドポイントを使用する場合、VPC と 間の通信 AWS KMS は AWS ネットワーク内で完全に行われます。

AWS KMS は、 を搭載した Amazon Virtual Private Cloud (Amazon VPC) エンドポイントをサポートしています[AWS PrivateLink](#)。各 VPC エンドポイントは、VPC サブネット内のプライベート IP アドレスを持つ 1 つ以上の [Elastic Network Interfaces](#) (ENI) で表されます。

インターフェイス VPC エンドポイントは、インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connect 接続 AWS KMS なしで VPC を直接 に接続します。VPC 内のインスタンスは、パブリック IP アドレスがなくても通信できます AWS KMS。

リージョン

AWS KMS は、[AWS KMS](#) がサポートされているすべての AWS リージョン で VPC エンドポイントと VPC エンドポイントポリシーをサポートします。

AWS KMS VPC エンドポイントに関する考慮事項

のインターフェイス VPC エンドポイントを設定する前に AWS KMS、「AWS PrivateLink ガイド」の「[インターフェイスエンドポイントのプロパティと制限](#)」トピックを確認してください。

AWS KMS VPC エンドポイントのサポートには以下が含まれます。

- VPC エンドポイントを使用して、VPC からすべての [AWS KMS API オペレーション](#) を呼び出すことができます。
- AWS KMS リージョンエンドポイントまたは [AWS KMS FIPS エンドポイント](#) に接続するインターフェイス VPC エンドポイントを作成できます。
- AWS CloudTrail ログを使用して、VPC エンドポイントを介した KMS キーの使用を監査できます。詳細については、「[VPC エンドポイントを使用する AWS KMS リクエストのログ記録](#)」を参照してください。

トピック

- [の VPC エンドポイントを作成する AWS KMS](#)
- [AWS KMS VPC エンドポイントに接続する](#)
- [VPC エンドポイントを使用して AWS KMS リソースへのアクセスを制御する](#)
- [VPC エンドポイントを使用する AWS KMS リクエストのログ記録](#)

の VPC エンドポイントを作成する AWS KMS

Amazon VPC コンソールまたは Amazon VPC API AWS KMS を使用して、の VPC エンドポイントを作成できます。次のいずれかの値を使用して、「[インターフェイスエンドポイントを作成する](#)」手順に従ってください。

- の VPC エンドポイントを作成するには AWS KMS、次のサービス名を使用します。

```
com.amazonaws.region.kms
```

例えば、米国西部 (オレゴン) リージョン (us-west-2) では、サービス名は次のようになります。

```
com.amazonaws.us-west-2.kms
```

- [AWS KMS FIPS エンドポイント](#) に接続する VPC エンドポイントを作成するには、次のサービス名を使用します。

```
com.amazonaws.region.kms-fips
```


例えば、米国西部 (オレゴン) リージョン (us-west-2) では、サービス名は次のようになります。

```
com.amazonaws.us-west-2.kms-fips
```

VPC エンドポイントを使いやすくするために、VPC エンドポイントに対して [プライベート DNS 名](#) を有効にすることができます。[Enable DNS Name] (DNS 名を有効にする) オプションを選択すると、標準の AWS KMS DNS ホスト名が VPC エンドポイントに解決されます。例えば、https://kms.us-west-2.amazonaws.com はサービス名 com.amazonaws.us-west-2.kms に接続された VPC エンドポイントに解決されます。

このオプションにより VPC エンドポイントが使いやすくなります。AWS SDKs とはデフォルトで標準の DNS AWS KMS ホスト名 AWS CLI を使用するため、アプリケーションとコマンドで VPC エンドポイント URL を指定する必要はありません。

詳細については、「AWS PrivateLink ガイド」の「[インターフェイスエンドポイントを介したサービスへアクセスする](#)」を参照してください。

AWS KMS VPC エンドポイントに接続する

AWS SDK、AWS CLI または を使用して、VPC エンドポイント AWS KMS を介して に接続できます AWS Tools for PowerShell。VPC エンドポイントを指定するには、DNS 名を使用します。

例えば、この [list-keys](#) コマンドは、endpoint-url パラメータを使用して VPC エンドポイントを指定します。こうしたコマンドを使用するには、サンプルの VPC エンドポイント ID を、ご自身のアカウントのものに置き換えてください。

```
$ aws kms list-keys --endpoint-url https://vpce-1234abcdef5678c90a-09p7654s-us-east-1a.ec2.us-east-1.vpce.amazonaws.com
```

必要なアクセス許可

VPC エンドポイントを使用する AWS KMS リクエストを成功させるには、プリンシパルに 2 つのソースからのアクセス許可が必要です。

- [キーポリシー](#)、[IAM ポリシー](#)、[権限](#) が リソース (KMS キーまたはエイリアス) でオペレーションを呼び出すために、プリンシパルにアクセス許可を付与する必要があります。
- VPC エンドポイントポリシーは、エンドポイントを使用してリクエストを実行するためのアクセス権限をプリンシパルに付与する必要があります。

例えば、キーポリシーがプリンシパルに、特定の KMS キーで [Decrypt](#) を呼び出すためのアクセス許可を付与します。ただし、VPC エンドポイントポリシーは、プリンシパルがエンドポイントを使用して、その KMS キーで Decrypt を呼び出すことを許可しない場合があります。

または、VPC エンドポイントポリシーは、プリンシパルがエンドポイントを使用して、特定の KMS キーで [DisableKey](#) を呼び出すことを許可する場合もあります。ただし、プリンシパルにキーポリシー、IAM ポリシー、または付与からのアクセス権限がない場合、リクエストは失敗します。

エンドポイントの作成時に VPC エンドポイントポリシーを作成できます。また、VPC エンドポイントポリシーはいつでも変更できます。VPC マネジメントコンソール、または [CreateVPcendPoint](#) オペレーションまたは [ModifyVPcendPoint](#) オペレーションを使用します。AWS CloudFormation テンプレートを使用して VPC エンドポイントポリシーを作成および変更することもできます。VPC マネジメントコンソールの使用方法については、「AWS PrivateLink ガイド」の「[インターフェイスエンドポイントの作成](#)」および「[インターフェイスエンドポイントの変更](#)」を参照してください。

プライベートホスト名

VPC エンドポイントの作成時にプライベートホスト名を有効にした場合は、CLI コマンドまたはアプリケーションの設定で VPC エンドポイント URL を指定する必要はありません。標準の AWS KMS DNS ホスト名は VPC エンドポイントに解決されます。AWS CLI および SDKs はデフォルトでこのホスト名を使用するため、VPC エンドポイントの使用を開始して、スクリプトやアプリケーションを変更せずに AWS KMS リージョンエンドポイントに接続できます。

プライベートホスト名を使用するには、VPC の `enableDnsHostnames` 属性と `enableDnsSupport` 属性を `true` に設定する必要があります。これらの属性を設定するには、[ModifyVpccattribute](#) オペレーションを使用します。詳細については、「Amazon VPC ユーザーガイド」の「[VPC の DNS 属性の表示と更新](#)」を参照してください。

VPC エンドポイントを使用して AWS KMS リソースへのアクセスを制御する

リクエストが VPC から送信されたとき、または VPC エンドポイントを使用する場合に、AWS KMS リソースとオペレーションへのアクセスを制御できます。このためには、[キーポリシー](#)または [IAM ポリシー](#)で、次のいずれかの[グローバル条件キー](#)を使用します。

- `aws:sourceVpce` 条件キーを使用して、VPC エンドポイントに基づいてアクセスを許可または制限します。

- `aws:sourceVpc` 条件キーを使用して、プライベートエンドポイントをホストする VPC に基づいてアクセスを許可または制限します。

Note

VPC エンドポイントに基づいてキーポリシーと IAM ポリシーを作成する場合は、注意が必要です。ポリシーステートメントで、特定の VPC または VPC エンドポイントからのリクエストが要求されている場合、ユーザーに代わって AWS KMS リソースを使用する統合 AWS サービスからのリクエストは失敗する可能性があります。ヘルプについては、「[AWS KMS アクセス許可を持つポリシーでの VPC エンドポイント条件の使用](#)」を参照してください。また、リクエストが [Amazon VPC エンドポイント](#) から送信される場合、`aws:sourceIP` 条件キーは無効です。リクエストを VPC エンドポイントに制限するには、`aws:sourceVpc` または `aws:sourceVpc` 条件キーを使用します。詳細については、「AWS PrivateLink ガイド」の「[VPC エンドポイントおよび VPC エンドポイントサービスの Identity and Access Management](#)」を参照してください。

これらのグローバル条件キーを使用して、AWS KMS keys (KMS キー)、エイリアス、および特定のリソースに依存しない [CreateKey](#) などのオペレーションへのアクセスを制御できます。

例えば、次のサンプルキーポリシーでは、リクエストが指定された VPC エンドポイントを使用する場合にのみ、KMS キーを使用してユーザーが一部の暗号化オペレーションを実行できます。ユーザーがリクエストを行うと AWS KMS、リクエストの VPC エンドポイント ID がポリシー `aws:sourceVpc` の条件キー値と比較されます。一致しない場合、要求は拒否されます。

このようなポリシーを使用するには、プレースホルダー AWS アカウント ID と VPC エンドポイント IDs をアカウントの有効な値に置き換えます。

JSON

```
{
  "Id": "example-key-1",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnableIAMpolicies",
      "Effect": "Allow",
      "Principal": {"AWS":["111122223333"]},
```

```
        "Action": ["kms:*"],
        "Resource": "*"
    },
    {
        "Sid": "Restrict usage to my VPC endpoint",
        "Effect": "Deny",
        "Principal": "*",
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*"
        ],
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "aws:sourceVpc": "vpce-1234abcdef5678c90a"
            }
        }
    }
}
```

また、aws:sourceVpc 条件キーを使用して、VPC エンドポイントが存在する VPC に基づいて、KMS キーへのアクセスを制限することもできます。

次のサンプルキーポリシーでは、コマンドが vpc-12345678 から送信される場合にのみ、KMS キーを管理するコマンドが許可されます。また、コマンドが vpc-2b2b2b2b から送信される場合にのみ、暗号化オペレーションで KMS キーを使用するコマンドが許可されます。ある VPC でアプリケーションが実行されていれば、このようなポリシーを使用できますが、管理機能のために 2 番目の切り離された VPC を使用します。

このようなポリシーを使用するには、プレースホルダー AWS アカウント ID と VPC エンドポイント IDs をアカウントの有効な値に置き換えます。

JSON

```
{
    "Id": "example-key-2",
    "Version": "2012-10-17",
```

```
"Statement": [  
  {  
    "Sid": "AllowAdministrativeActionsFromVPC",  
    "Effect": "Allow",  
    "Principal": {  
      "AWS": "111122223333"  
    },  
    "Action": [  
      "kms:Create*",  
      "kms:Enable*",  
      "kms:Put*",  
      "kms:Update*",  
      "kms:Revoke*",  
      "kms:Disable*",  
      "kms>Delete*",  
      "kms:TagResource",  
      "kms:UntagResource"  
    ],  
    "Resource": "*",  
    "Condition": {  
      "StringEquals": {  
        "aws:sourceVpc": "vpc-12345678"  
      }  
    }  
  },  
  {  
    "Sid": "AllowKeyUsageFromVPC2b2b2b2b",  
    "Effect": "Allow",  
    "Principal": {  
      "AWS": "111122223333"  
    },  
    "Action": [  
      "kms:Encrypt",  
      "kms:Decrypt",  
      "kms:GenerateDataKey*"  
    ],  
    "Resource": "*",  
    "Condition": {  
      "StringEquals": {  
        "aws:sourceVpc": "vpc-2b2b2b2b"  
      }  
    }  
  },  
  {  

```

```
    "Sid": "AllowReadActionsEverywhere",
    "Effect": "Allow",
    "Principal": {
        "AWS": "111122223333"
    },
    "Action": [
        "kms:Describe*",
        "kms:List*",
        "kms:Get*"
    ],
    "Resource": "*"
  }
]
```

VPC エンドポイントを使用する AWS KMS リクエストのログ記録

AWS CloudTrail は、VPC エンドポイントを使用するすべてのオペレーションを記録します。へのリクエストが VPC エンドポイント AWS KMS を使用する場合、VPC エンドポイント ID はリクエストを記録する[AWS CloudTrail ログ](#)エントリに表示されます。エンドポイント ID を使用して、AWS KMS VPC エンドポイントの使用を監査できます。

ただし、CloudTrail ログには、他のアカウントのプリンシパルによってリクエストされたオペレーションや、他のアカウントの KMS キーとエイリアスに対する AWS KMS オペレーションのリクエストは含まれません。また、お客様の VPC 保護のため、VPC エンドポイントポリシーによって拒否されたが、それ以外の場合には許可されていたであろうと思われるリクエストは [AWS CloudTrail](#) には記録されません。

例えば、このサンプルログエントリは、VPC エンドポイントを使用した [GenerateDataKey](#) リクエストを記録します。vpcEndpointId フィールドは、ログエントリの最後に表示されます。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "accountId": "111122223333",
    "userName": "Alice"
  },
```

```
{
  "eventTime": "2018-01-16T05:46:57Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "eu-west-1",
  "sourceIPAddress": "172.01.01.001",
  "userAgent": "aws-cli/1.14.23 Python/2.7.12 Linux/4.9.75-25.55.amzn1.x86_64
botocore/1.8.27",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "numberOfBytes": 128
  },
  "responseElements": null,
  "requestID": "a9fff0bf-fa80-11e7-a13c-afcabbff2f04c",
  "eventID": "77274901-88bc-4e3f-9bb6-acf1c16f6a7c",
  "readOnly": true,
  "resources": [{
    "ARN": "arn:aws:kms:eu-west-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333",
    "type": "AWS::KMS::Key"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333",
  "vpceEndpointId": "vpce-1234abcd5678c90a"
}
```

デュアルスタックエンドポイントのサポート

AWS KMS は、IPv4 クライアントと IPv6 クライアントの両方をサポートするデュアルスタックのパブリックエンドポイントを提供します。デュアルスタックエンドポイントを使用すると、クライアントは IPv4 アドレスまたは IPv6 アドレス AWS KMS を使用してと通信できます。AWS KMS エンドポイントの詳細については、[AWS Key Management Service 「エンドポイントとクォータ」を参照してください。](#)

の AWS KMS デュアルスタックパブリックエンドポイントは、IPv4 クライアントと IPv6 クライアントの両方 `https://kms.your-region.api.aws` をサポートします。AWS KMS は、を使用して仮想プライベートクラウド (VPC) から IPv4 および IPv6 経由でプライベートにアクセスすることもできます AWS PrivateLink。のプライベートインターフェイス VPC エンドポイントの作成の詳細については AWS KMS、[「」を参照してください](#) [VPC エンドポイント AWS KMS を介してに接続する。](#)

VPC の IPv6 アドレス指定の詳細については、[「Amazon Virtual Private Cloud ユーザーガイド」の「Amazon VPC の仕組み」](#)を参照してください。VPCs Amazon Virtual Private Cloud VPC をデュアルスタックモードに設定する方法の詳細については、Amazon Virtual Private Cloud ユーザーガイド[VPCs とサブネットの IP アドレス指定](#)」を参照してください。

IPv6 で使用できない機能

AWS KMS は IPv6 経由で AWS CloudHSM キーストアまたは外部キーストアと通信できません。この制限は、IPv6 経由で AWS KMS APIs呼び出すことを妨げません。

AWS KMS の概念

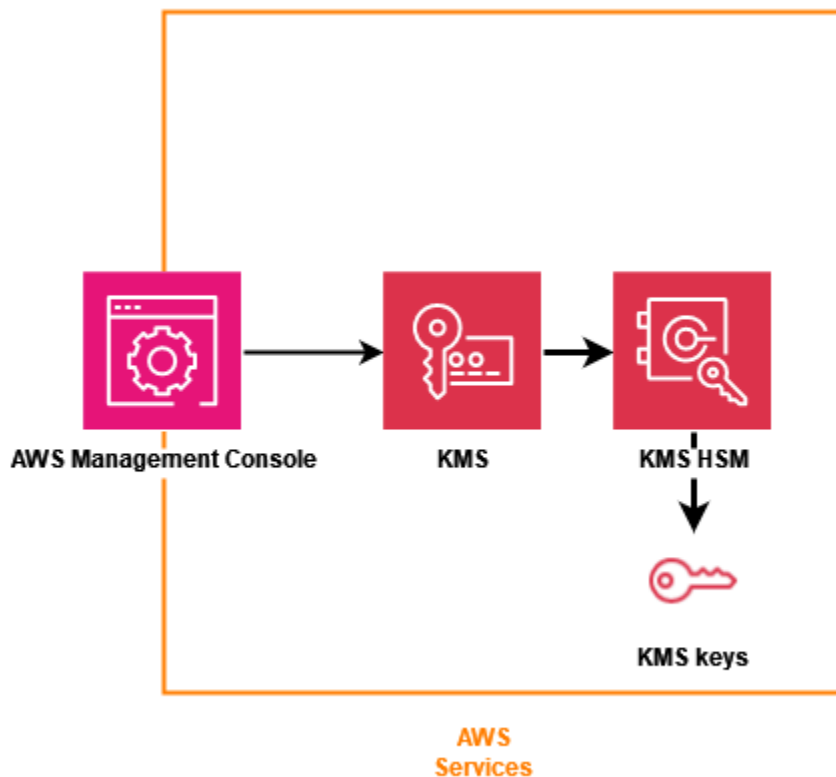
AWS Key Management Service (AWS KMS) で使用される基本的な用語と概念、およびそれらが連携してデータを保護する方法について説明します。

の概要 AWS KMS

AWS Key Management Service (AWS KMS) は、暗号化キーを生成および管理するためのウェブインターフェイスを提供し、データを保護するための暗号化サービスプロバイダーとして機能します。は、AWS サービスと統合された従来のキー管理サービス AWS KMS を提供し AWS、一元的な管理と監査により、全体で顧客のキーの一貫したビューを提供します。

AWS KMS には AWS Management Console、を介したウェブインターフェイス、コマンドラインインターフェイス、および RESTful API オペレーションが含まれており、FIPS 140-3 検証済みハードウェアセキュリティモジュール (HSMs)。AWS KMS HSM は、のセキュリティとスケーラビリティの要件を満たす専用の暗号化機能を提供するように設計されたマルチチップスタンドアロンハードウェア暗号化アプライアンスです AWS KMS。AWS KMS keysとして管理しているキーで、独自の HSM ベースの暗号化階層を確立できます。これらのキーは HSM 上でのみ使用でき、暗号化リクエストの処理に必要な時間だけメモリ内にあります。複数の KMS キーを作成できます。各キーはキー ID で表されます。各顧客が管理する AWS IAM ロールとアカウントでのみ、カスターマネージド KMS キーを作成、削除、またはデータの暗号化、復号、署名、検証に使用できます。キーにアタッチされたポリシーを作成することで、KMS キーを管理および/または使用できるユーザーに関するアクセス制御を定義できます。このようなポリシーを使用すると、API オペレーションごとにキーにアプリケーション固有の使用を定義できます。

さらに、ほとんどの AWS サービスは、KMS キーを使用した保管中のデータの暗号化をサポートしています。この機能を使用すると、KMS キーにアクセスする方法とタイミングを制御することで、AWS サービスが暗号化されたデータにアクセスする方法とタイミングを制御できます。



AWS KMS は、ウェブ向け AWS KMS ホストと HSMs。これらの階層型ホストのグループ化は AWS KMS スタックを形成します。へのすべてのリクエストは、Transport Layer Security プロトコル (TLS) 経由で行い、AWS KMS ホストで終了 AWS KMS する必要があります。AWS KMS ホストは、完全な[前方秘匿性](#)を提供する暗号スイートでのみ TLS を許可します。AWS KMS は、他のすべての AWS API オペレーションで利用できる AWS Identity and Access Management (IAM) の同じ認証情報とポリシーメカニズムを使用してリクエストを認証および認可します。

AWS KMS 設計目標

AWS KMS は、以下の要件を満たすように設計されています。

耐久性

暗号化キーの耐久性は、の最高の耐久性サービスに等しいように設計されています AWS。1 つの暗号化キーで、長期間にわたって蓄積された大量のデータを暗号化できます。

信頼性

キーの使用は、ユーザーが定義および管理するアクセス制御ポリシーによって保護されます。プレーンテキストの KMS キーをエクスポートするメカニズムはありません。暗号化キーの機密性

は重要です。HSM で管理アクションを実行するには、定足数ベースのアクセス制御にロール固有のアクセス権を持つ複数の Amazon 従業員が必要です。

低レイテンシーと高スループット

AWS KMS は、の他のサービスでの使用に適したレイテンシーとスループットレベルで暗号化オペレーションを提供します AWS。

リージョンの独立性

AWS は、異なるリージョンでデータアクセスを制限する必要があるお客様向けに、独立したリージョンを提供します。キーの使用は、AWS リージョン内に限ることができます。

元になる乱数の安全性

強力な暗号化は、真に予測不可能な乱数生成に依存するため、は、高品質で検証済みの乱数のソース AWS KMS を提供します。

監査

AWS KMS は、AWS CloudTrail ログに暗号化キーの使用と管理を記録します。AWS CloudTrail ログを使用して、ユーザーに代わって AWS のサービスによるキーの使用など、暗号化キーの使用を検査できます。

これらの目標を達成するために、AWS KMS システムには、「ドメイン」を管理する AWS KMS 一連の演算子とサービスホスト演算子 (総称して「オペレータ」) が含まれています。ドメインは、リージョンで定義された AWS KMS サーバー、HSMs、および演算子のセットです。各 AWS KMS 演算子には、アクションの認証に使用されるプライベートキーとパブリックキーのペアを含むハードウェアトークンがあります。HSM には、HSM 状態の同期を保護する暗号化キーを確立するために、追加のプライベートとパブリックのキーペアがあります。

AWS KMS keys

お客様独自の暗号化アプリケーションで使用するために作成および管理している KMS キーは、カスタマーマネージドキーと呼ばれるタイプのキーです。カスタマーマネージドキーは、KMS キーを使用して AWS サービスがユーザーに代わって保存するデータを暗号化するサービスと組み合わせて使用することもできます。カスタマーマネージドキーは、使用するキーのライフサイクルと用途を完全にコントロールしたいお客様にお勧めします。アカウント内でカスタマーマネージドキーを使用する場合、月額料金が発生します。さらに、キーの使用および管理のリクエストにも、使用コストが発生します。詳細については、「[AWS Key Management Service の料金](#)」を参照してください。

顧客が AWS サービスでデータを暗号化したいが、キー管理のオーバーヘッドを望まず、キーの支払いを望まない場合があります。AWS マネージドキー は、お客様のアカウント内に存在する KMS キーで、特定の状況でのみ使用できます。具体的には、運用している AWS サービスのコンテキストでのみ使用でき、キーが存在するアカウント内のプリンシパルのみが使用できます。これらのキーのライフサイクルやアクセス許可については、お客様は一切管理することができません。AWS サービスで暗号化機能を使用すると、「aws<service code> AWS マネージドキー」という形式のエイリアスを使用する場合があります。例えば、aws/ebs キーは EBS ボリュームの暗号化にのみ使用でき、キーと同じアカウントの IAM プリンシパルが使用するボリュームにのみ使用できます。アカウント内のリソースについて、アカウントのユーザーのみが使用できるようにスコープダウン AWS マネージドキー されている を考えてみましょう。で暗号化されたリソース AWS マネージドキー を他のアカウントと共有することはできません。AWS マネージドキー はアカウントに無料で存在できますが、このキータイプの使用は、キーに割り当てられた AWS サービスによって課金されます。

AWS マネージドキー は、2021 年の時点で新しい AWS サービス用に作成されなくなったレガシーキータイプです。代わりに、新しい (およびレガシー) AWS 所有のキー AWS サービスは、と呼ばれるものを使用して、デフォルトで顧客データを暗号化します。は、AWS サービスによって管理されるアカウントにある KMS AWS 所有のキー キーであるため、サービスオペレーターはライフサイクルと使用のアクセス許可を管理できます。を使用することで AWS 所有のキー、AWS サービスはデータを透過的に暗号化し、キーアクセス許可を心配することなく、クロスアカウントまたはクロスリージョンで簡単にデータを共有できます。AWS 所有のキー は、よりシンプルで自動化されたデータ保護を提供するデフォルト暗号化ワークロードとしてご利用いただけます。これらのキーはによって所有および管理されるため AWS、その存在や使用状況に対して課金されず、ポリシーを変更したり、これらのキーのアクティビティを監査したり、削除したりすることはできません。コントロールを重視される場合にはカスタマーマネージドキーを使用し、利便性を重視される場合は AWS 所有のキー をご利用ください。

	カスタマーマネージドキー	AWS マネージドキー	AWS 所有のキー
キーポリシー	カスタマーが排他的に制御する	サービスによって制御され、カスタマーは閲覧可能	排他的に制御され、データを暗号化する AWS サービスでのみ表示可能
ログ記録	CloudTrail カスタマー証跡またはイベントデータストア	CloudTrail カスタマー証跡またはイベントデータストア	カスタマーは閲覧できない

ライフサイクル管理	カスタマーがローテーション、削除、リージョンロケーションを管理する	AWS KMS ローテーション (年間)、削除、リージョンロケーションを管理します	AWS のサービス ローテーション、削除、リージョンロケーションを管理します
料金表	キーの存在に対する月額料金 (時間単位で按分)。キーの使用にも課金される	月額料金はかかりませんが、キーの API 使用に対して各呼び出し元に課金されます。	カスタマーへの請求はなし

お客様が作成する KMS キーは[カスタマーマネージドキー](#)です。サービスリソースの暗号化に KMS キーを使用する AWS のサービス では、多くの場合お客様が使うキーを作成します。AWS アカウントで AWS のサービス 作成する KMS キーは [AWS マネージドキー](#)。がサービスアカウントで AWS のサービス 作成する KMS キーは [AWS 所有のキー](#)。

KMS キーのタイプ	KMS キーメタデータを表示可能	KMS キーを管理可能	自分の のみ使用する AWS アカウント	自動ローテーション	料金表
カスタマーマネージドキー	はい	あり	はい	オプション。	月額料金 (時間単位の日割り計算) 従量課金料金
AWS マネージドキー	はい	なし	はい	必須。毎年 (約 365 日)	月額料金なし 使用ごとの料金 (一部のは、この料金を AWS のサービス 支払います)

KMS キーの タイプ	KMS キーメ タデータを表 示可能	KMS キーを 管理可能	自分の のに み使用する AWS アカウ ント	自動ローテー ション	料金表
AWS 所有の キー	いいえ	いいえ	いいえ	はローテー ション戦略 AWS のサー ビス を管理 します。	料金はかかり ません

[AWS と統合する サービスは AWS KMS](#)、KMS キーのサポートが異なります。一部の AWS サービスは、デフォルトで AWS 所有のキー または を使用してデータを暗号化します AWS マネージドキー。一部の AWS サービスは、カスタマーマネージドキーをサポートしています。他の AWS サービスは、 の使いやすさ AWS 所有のキー、 の可視性 AWS マネージドキー、またはカスタマーマネージドキーの制御を可能にするすべてのタイプの KMS キーをサポートしています。AWS サービスが提供する暗号化オプションの詳細については、「ユーザーガイド」の「保管時の暗号化」トピックまたは サービスの開発者ガイドを参照してください。

カスタマーマネージドキー

お客様が作成する KMS キーは、カスタマーマネージドキーです。カスタマーマネージドキーは、AWS アカウント ユーザーが作成、所有、管理する の KMS キーです。これらの KMS キーは、[キーポリシー](#)、[IAM ポリシー](#)、[グラント](#)の確立と管理、[有効化と無効化](#)、[暗号化マテリアルのローテーション](#)、[タグの追加](#)、KMS キーを参照する[エイリアスの作成](#)、[KMS キー削除のスケジューリング](#)などを完全に制御できます。

カスタマーマネージドキーは、AWS KMSの AWS Management Console のカスタマーマネージドキーページに表示されます。カスタマーマネージドキーを明確に識別するには、[DescribeKey](#) オペレーションを使用します。カスタマーマネージドキーでは、DescribeKey レスポンスの KeyManager フィールドの値は CUSTOMER です。

暗号化オペレーションでカスタマーマネージドキーを使用し、AWS CloudTrail ログでその使用を監査できます。さらに、[AWS KMSと統合されている多数のAWS のサービス](#)を使用すると、カスタマーマネージドキーを指定して保存および管理するデータを保護できます。

カスタマーマネージドキーの使用には、月額料金と、無料利用枠を超えた使用に対する料金がかかります。アカウントの AWS KMS [クォータ](#) に対してカウントされます。詳細については、「[AWS Key Management Service 料金表](#)」および「[クォータ](#)」を参照してください。

AWS マネージドキー

AWS マネージドキー は、アカウント内の KMS キーであり、[AWS と統合された サービス AWS KMS](#)によってユーザーに代わって作成、管理、使用されます。

一部の AWS サービスでは、AWS マネージドキー またはカスタマーマネージドキーを選択して、そのサービスのリソースを保護できます。一般的に、リソースを保護する暗号化キーを制御する必要があります。AWS マネージドキー が適しています。キーまたはそのキーポリシーを作成または維持する必要はなく、AWS マネージドキーの月額料金はまったく発生しません。

アカウントで [を表示 AWS マネージドキー](#)し、[そのキーポリシーを表示](#)し、AWS CloudTrail ログでの使用 [を監査](#)するアクセス許可があります。ただし、のプロパティの変更 AWS マネージドキー、ローテーション、キーポリシーの変更、削除のスケジュールはできません。また、暗号化オペレーション AWS マネージドキー で直接使用することはできません。それらを作成するサービスは、ユーザーに代わってそれらを使用します。

組織の[リソースコントロールポリシー](#)は適用されません AWS マネージドキー。

AWS マネージドキー AWS Management Console for の AWS マネージドキーページに表示されます AWS KMS。aws/*service-name*などの形式を持つエイリアス AWS マネージドキー でを識別することもできますaws/redshift。を明確に識別するには AWS マネージドキー、[DescribeKey](#) オペレーションを使用します。AWS マネージドキーの場合、DescribeKey レスポンスの KeyManager フィールドの値は AWS です。

すべての AWS マネージドキー は毎年自動的にローテーションされます。このローテーションスケジュールは変更できません。

Note

2022 年 5 月、のローテーションスケジュールを 3 年ごと (約 1,095 日) AWS マネージドキー から 1 年ごと (約 365 日) AWS KMS に変更しました。

の月額料金はかかりません AWS マネージドキー。無料利用枠を超えると使用料が発生する場合がありますが、一部の AWS サービスではこれらのコストがかかります。詳細については、サービスの

ユーザーガイドまたはデベロッパーガイドの「保管時の暗号化」トピックを参照してください。詳細については、「[AWS Key Management Service の料金表](#)」を参照してください。

AWS マネージドキー は、アカウントの各リージョンの KMS キー数のリソースクォータにはカウントされません。ただし、アカウントのプリンシパルに代わって使用される場合、KMS キーはリクエストクォータに対してカウントされます。詳細については、「[クォータ](#)」を参照してください。

AWS 所有のキー

AWS 所有のキー は、AWS サービスが複数の で使用するために所有および管理する KMS キーのコレクションです AWS アカウント。AWS 所有のキー は にはありませんが AWS アカウント、AWS サービスは AWS 所有のキー を使用して アカウントのリソースを保護できます。

一部の AWS サービスでは、AWS 所有のキー またはカスターマネージドキーを選択できます。一般的に、 リソースを保護する暗号化キーを監査または制御する必要がない限り、AWS 所有のキー は適切な選択です。AWS 所有のキー は完全に無料 (月額料金や使用料なし) であり、アカウントの [AWS KMS クォータ](#) にはカウントされず、簡単に使用できます。キーまたはそのキーポリシーを作成または管理する必要はありません。

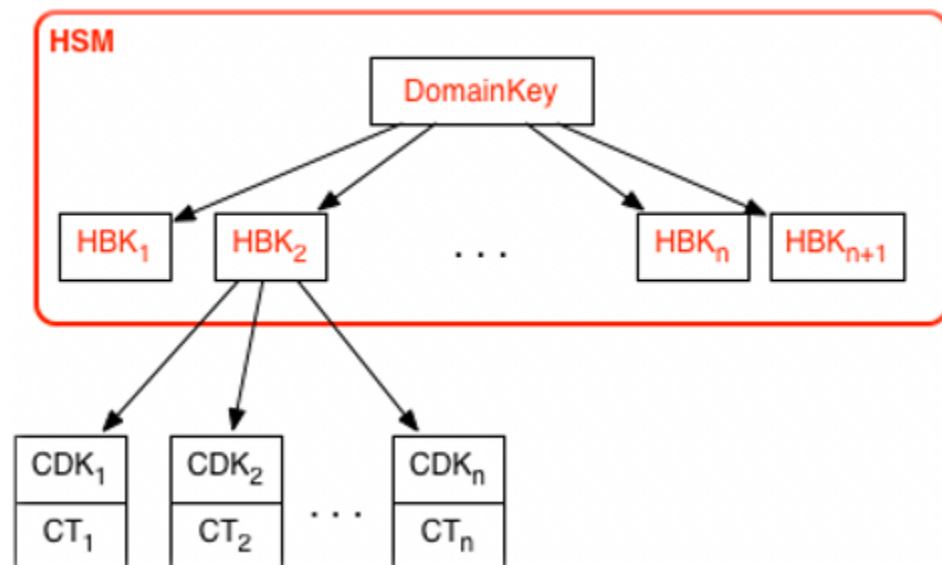
のローテーションはサービス AWS 所有のキー によって異なります。特定の のローテーションの詳細については AWS 所有のキー、 サービスのユーザーガイドまたは開発者ガイドの「保管時の暗号化」トピックを参照してください。

AWS KMS key 階層

キー階層は、最上位の論理キー で始まります AWS KMS key。KMS キーは、最上位のキーマテリアルのコンテナを表し、Amazon リソースネーム (ARN) を持つ AWS のサービス名前空間内で一意に定義されます。ARN には、一意に生成されたキー識別子である キー ID が含まれています。KMS キーは、ユーザーが開始したリクエストに基づいて作成されます AWS KMS。受信時に、 は KMS キーコンテナに配置する最初の HSM バックアップキー (HBK) の作成を AWS KMS リクエストします。HBK は、ドメイン内の HSM 上で生成され、プレーンテキストで HSM からエクスポートされないように設計されています。そうではなく、HBK は HSM 管理ドメインキーで暗号化されてエクスポートされます。これらのエクスポートされた HBK は、エクスポートされたキートークン (EKT) と呼ばれます。

EKT は、耐久性に優れた低レイテンシーのストレージにエクスポートされます。例えば、論理 KMS キーへの ARN を受け取ったとします。これは、キー階層、つまり暗号化コンテキストの最上位を表します。アカウント内に複数の KMS キーを作成し、他の AWS 名前付きリソースと同様に KMS キーにポリシーを設定できます。

特定の KMS キーの階層内では、HBK は KMS キーのバージョンと考えることができます。KMS キーをローテーションする場合は AWS KMS、新しい HBK が作成され、KMS キーのアクティブな HBK として KMS キーに関連付けられます。古い HBK は保持され、以前に保護されていたデータの復号化と検証に使用できます。ただし、新しい情報を保護するために使用できるのはアクティブな暗号化キーだけです。



AWS KMS を使用して KMS キーを使用して情報を直接保護したり、KMS キーで保護されている追加の HSM 生成キーをリクエストしたりできます。これらのキーは、お客様データキー、または CDK と呼ばれます。CDK は、暗号文 (CT) として暗号化して、プレーンテキストで、またはその両方で返すことができます。KMS キー (お客様が用意したデータまたは HSM が生成したキー) で暗号化されたすべてのオブジェクトは、呼び出しを介して HSM でのみ復号できます AWS KMS。

返された暗号文、または復号されたペイロードが 内に保存されることはありません AWS KMS。この情報は、AWS KMS への TLS 接続を介してユーザーに直接返されます。これは、ユーザーに代わって AWS のサービスによって行われた呼び出しにも適用されます。

次の表に、キーの階層と特定のキープロパティを示します。

キー	説明	ライフサイクル
ドメインキー	HSM のメモリ内のみにある 256 ビットの AES-GCM キー。KMS キーのバージョン (HSM バックアップキー) をまとめるために使用されます。	毎日ローテーション ¹

キー	説明	ライフサイクル
HSM バックアップキー	256 ビットの対称キー、または、RSA または楕円曲線のプライベートキー。顧客データとキーを保護するために使用され、ドメインキーの下に暗号化されて保存されます。1 つ以上の HSM バックアップキーで、keyId で表される KMS キーが構成されます。	毎年ローテーション ² (設定はオプション)。
派生暗号化キー	お客様データとキーを暗号化するために使用される HSM のメモリ内のみにある 256 ビット AES-GCM キー。暗号化ごとに HBK から派生します。	暗号化ごとに 1 回使用され、復号時に再生成されます
お客様データキー	プレーンテキストおよび暗号文で HSM からエクスポートされたユーザー定義の対称キーまたは非対称キー。 HSM バックアップキーで暗号化され、TLS チャネル経由で許可されたユーザーに返されます。	アプリケーションによって制御されるローテーションと使用

¹ AWS KMS ドメインの管理および設定タスクを考慮して、ドメインキーのローテーションを最大週 1 回まで緩和する場合があります。

² AWS KMS ユーザーに代わって、によって AWS マネージドキー 作成および管理されるデフォルトは、毎年自動的にローテーションされます。

キー識別子 (KeyId)

キー識別子は、KMS キーの名前のように機能します。これらは、コンソールで KMS キーを識別するのに役立ちます。キー識別子を使用して、AWS KMS API オペレーション、キーポリシー、IAM ポリシー、グラントで使用する KMS キーを指定します。キー識別子の値は、KMS キーに関連付けられているキーマテリアルとはまったく関係ありません。

AWS KMS は、複数のキー識別子を定義します。KMS キーを作成すると、は KMS キーのプロパティであるキー ARN とキー ID AWS KMS を生成します。[エイリアス](#)を作成すると、は定義したエイリアス名に基づいてエイリアス ARN AWS KMS を生成します。キー識別子とエイリアス識別子は、AWS Management Console および AWS KMS API で表示できます。

AWS KMS コンソールでは、キー ARN、キー ID、またはエイリアス名で KMS キーを表示およびフィルタリングし、キー ID とエイリアス名でソートできます。コンソールでキー ID を検索する方法については、「[the section called “キー ID とキー ARN を検索する”](#)」を参照してください。

AWS KMS API では、KMS キーを識別するために使用するパラメータの名前、KeyIdまたはTargetKeyIdやなどのバリエーションが付けられますDestinationKeyId。ただし、これらのパラメータの値はキー ID に限定されません。いくつかは、任意の有効なキー識別子を受け取ることができます。各パラメータの値については、AWS Key Management Service API リファレンスのパラメータの説明を参照してください。

Note

AWS KMS API を使用する場合は、使用するキー識別子に注意してください。API ごとに異なるキー識別子が必要です。通常、タスクに対して実用的で最も完全なキー識別子を使用します。

AWS KMS は、次のキー識別子をサポートしています。

キー ARN

キー ARN は、KMS キーの Amazon リソースネーム (ARN) です。これは KMS キーの、一意の完全修飾識別子です。キー ARN には AWS アカウント、、リージョン、およびキー ID が含まれます。KMS キーのキー ARN を検索する方法については、[the section called “キー ID とキー ARN を検索する”](#) を参照してください。

キー ARN の形式は次のとおりです。

```
arn:<partition>:kms:<region>:<account-id>:key/<key-id>
```

以下は単一リージョン KMS キーのサンプルキー ARN です。

```
arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

#####のキー ARN の[key-id](#) 要素は、mrk- プレフィックスで始まります。以下はマルチリージョンキーのサンプルキー ARN です。

```
arn:aws:kms:us-west-2:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab
```

キー ID

キー ID は、アカウントとリージョン内の KMS キーを一意に識別します。KMS キーのキー ID を検索する方法については、[the section called “キー ID とキー ARN を検索する”](#) を参照してください。

以下は単一リージョン KMS キーのサンプルキー ID です。

```
1234abcd-12ab-34cd-56ef-1234567890ab
```

[マルチリージョンキー](#)のキー ID は、mrk- プレフィックスで始まります。以下はマルチリージョンキーのサンプルキー ID です。

```
mrk-1234abcd12ab34cd56ef1234567890ab
```

エイリアス ARN

エイリアス ARN は、エイリアスの Amazon リソースネーム (ARN) AWS KMS です。これは、エイリアスとそれが表す KMS キーの、一意の完全修飾識別子です。エイリアス ARN には AWS アカウント、、リージョン、エイリアス名が含まれます。

エイリアス ARN は、任意の時点で特定の 1 つの KMS キーを識別します。ただし、エイリアスに関連付けられている KMS キーは変更できるため、エイリアス ARN は異なる時点でさまざまな KMS キーを識別できます。KMS キーのエイリアス ARN を検索する方法については、[KMS キーのエイリアス名とエイリアス ARN を見つける](#) を参照してください。

エイリアス ARN の形式は次のとおりです。

```
arn:<partition>:kms:<region>:<account-id>:alias/<alias-name>
```

以下は、架空の ExampleAlias のエイリアス ARN です。

```
arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias
```

エイリアス名

エイリアス名は、最大 256 文字の文字列です。エイリアス名は、アカウントとリージョン内で関連付けられた KMS キーを一意に識別します。AWS KMS API では、エイリアス名は常に `alias/` で始まります。KMS キーのエイリアス名を検索する方法については、[KMS キーのエイリアス名とエイリアス ARN を見つける](#) を参照してください。

エイリアス名の形式は次のとおりです。

```
alias/<alias-name>
```

例:

```
alias/ExampleAlias
```

エイリアス名の `aws/` プレフィックスは、[AWS マネージドキー](#) 用に予約されます。このプレフィックスでエイリアスを作成することはできません。例えば、Amazon Simple Storage Service (Amazon S3) [AWS マネージドキー](#) のエイリアス名は次のとおりです。

```
alias/aws/s3
```

の非対称キー AWS KMS

非対称 KMS キーは、数学的に関連する公開キーとプライベートキーペアを表します。パブリックキーは、たとえ信頼されていなくても、誰にでも渡すことができますが、シークレットキーは秘密にしておく必要があります。

非対称 KMS キーでは、プライベートキーは作成 AWS KMS され、暗号化されていないままになる AWS KMS ことはありません。プライベートキーを使用するには、`GetPrivateKey` を呼び出す必要があります AWS KMS。API `GetPrivateKey` AWS KMS オペレーションを呼び出す AWS KMS ことで、内でパブリックキーを使用できます。または、[パブリックキーをダウンロード](#)して、の外部で使用できます AWS KMS。

`GetPublicKey` を呼び出すことができないユーザーがの外部 AWS で暗号化を必要とするユースケースでは AWS KMS、非対称 KMS キーが適しています。ただし、AWS サービスに保存または管理するデータを暗号化する KMS キーを作成する場合は、対称暗号化 KMS キーを使用します。[AWS と統合されたサービスは AWS KMS](#)、対称暗号化 KMS キーのみを使用してデータを暗号化します。これらのサービスは、非対称 KMS キーを使用する暗号化をサポートしません。

AWS KMS は 3 種類の非対称 KMS キーをサポートしています。

RSA KMS キー

暗号化と復号、または署名と検証用の RSA キーペアを持つ KMS キー (両方ではありません)。は、さまざまなセキュリティ要件に対して複数のキー長 AWS KMS をサポートします。

が RSA KMS キーに対して AWS KMS サポートする暗号化および署名アルゴリズムの技術的な詳細については、[「RSA キー仕様」](#)を参照してください。

楕円曲線 (ECC) KMS キー

署名と検証、または共有シークレットを取得するための楕円曲線キーペアを持つ KMS キー (両方ではありません)。は、一般的に使用される複数の曲線 AWS KMS をサポートします。

が ECC KMS キーで AWS KMS サポートする署名アルゴリズムの技術的な詳細については、[「楕円曲線のキー仕様」](#)を参照してください。

ML-DSA KMS キー

署名と検証用の ML-DSA キーペアを持つ KMS キー。ML-DSA は、量子コンピューティングによってもたらされるセキュリティの脅威から保護するために米国国立標準技術研究所 (NIST) によって開発されたポスト量子暗号標準です。ML-DSA は、RSA または楕円曲線デジタル署名アルゴリズムからポスト量子セーフ暗号に移行する組織に推奨されるデジタル署名アルゴリズムです。

AWS KMS は、さまざまなセキュリティ要件に対して複数のキー長をサポートします。が ML-DSA KMS キーで AWS KMS サポートする署名アルゴリズムの技術的な詳細については、[「ML-DSA キー仕様」](#)を参照してください。

SM2 KMS キー仕様 (中国リージョンのみ)

暗号化と復号、署名と検証、共有シークレットの取得のための SM2 キーペアを持つ KMS キー (1 つの[Key usage](#)タイプを選択する必要があります)。

が SM2 KMS キー (中国リージョンのみ) で AWS KMS サポートする暗号化および署名アルゴリズムの技術的な詳細については、[SM2 キー仕様](#)」を参照してください。

非対称キーの設定を選択する方法については、[「作成する KMS キーのタイプの選択」](#)を参照してください。

リージョン

非対称 KMS キーと非対称データキーペアは、AWS KMS がサポート AWS リージョン するすべてのリージョンでサポートされています。

詳細はこちら

- 非対称 KMS キーを作成するには、「[非対称 KMS キーを作成する](#)」を参照してください。
- マルチリージョンの非対称 KMS キーを作成するには、「[マルチリージョンのプライマリキーを作成する](#)」を参照してください。
- 非対称 KMS キーを使用してメッセージに署名し、署名を検証する方法については、AWS セキュリティブログの [Digital signing with the new asymmetric keys feature of AWS KMS](#) を参照してください。
- 非対称 KMS キーの削除に関する注意事項については、「[Deleting asymmetric KMS keys](#)」を参照してください。
- 非対称 KMS キーの特定と閲覧については、「[非対称 KMS キーを特定する](#)」を参照してください。

の HMAC キー AWS KMS

Hash-based Message Authentication Code (HMAC) KMS キーは、AWS KMS内で HMAC を生成して検証するために使用される対称キーです。それぞれの HMAC KMS キーに関連する一意のキーマテリアルは、HMAC アルゴリズムが必要とするシークレットキーを提供します。HMAC KMS キーを [GenerateMac](#) および [VerifyMac](#) オペレーションで使用して、AWS KMS内のデータの整合性と信頼性を検証することができます。

HMAC アルゴリズムは、暗号化ハッシュ関数と共有シークレットキーを組み合わせます。これらはメッセージとシークレットキー (HMAC KMS キーのキーマテリアルなど) を使用して、一意の固定サイズのコードまたはタグを返します。メッセージの文字が 1 字でも違う場合、またはシークレットキーが同一ではない場合、結果として得られるタグはまったく異なるものになります。HMAC は、シークレットキーをリクエストすることによって信頼性も提供します。シークレットキーがなければ、同一の HMAC タグを生成することは不可能です。HMAC は対称署名と呼ばれることもあります。これらはデジタル署名のように機能しますが、署名と検証の両方に単一のキーを使用するからです。

が AWS KMS 使用する HMAC KMS キーと HMAC アルゴリズムは、[RFC 2104](#) で定義されている業界標準に準拠しています。AWS KMS [GenerateMac](#) オペレーションは、標準の HMAC タグを生成します。HMAC KMS キーは、[FIPS 140-3 暗号化モジュール検証プログラム](#) (中国 (北京) および中国 (寧夏) リージョンを除く) で認定された AWS KMS ハードウェアセキュリティモジュールで生成さ

れ、暗号化 AWS KMS されていないままにされることはありません。HMAC KMS キーを使用するには、AWS KMS を呼び出す必要があります。

HMAC を使用して、JSON Web トークン (JWT)、トークン化されたクレジットカード情報、または送信されたパスワードなどのメッセージの信頼性を判断することができます。決定論的なキーを必要とするアプリケーションでは特に、セキュアなキー導出関数 (KDF) としても使用することもできます。

HMAC KMS キーは、アプリケーションソフトウェアからの HMACs よりも利点があります。キーマテリアルは、キーに設定したアクセスコントロールに従って AWS KMS、完全に内部で生成および使用されるためです。

Tip

ベストプラクティスでは、HMAC を含めたどの署名メカニズムについても、その有効時間を制限することが推奨されています。そうすることで、アクターが有効性を何度も確立したり、メッセージが置き換えられた後も長期間有効性を確立したりする攻撃が阻止されます。HMAC タグにタイムスタンプは含まれませんが、トークンまたはメッセージにタイムスタンプを含めて、HMAC を更新するタイミングを検知できるようにすることが可能です。

サポートされている暗号化オペレーション

HMAC KMS キーは、[GenerateMac](#) および [VerifyMac](#) 暗号化オペレーションのみをサポートします。HMAC KMS キーを使用してデータの暗号化やメッセージの署名を行ったり、HMAC オペレーションで他のタイプの KMS キーを使用したりすることはできません。GenerateMac オペレーションを使用するときは、ユーザーが最大 4,096 バイトのメッセージ、HMAC KMS キー、および HMAC キー仕様との互換性がある MAC アルゴリズムを指定して、GenerateMac が HMAC タグを計算します。HMAC タグを検証するには、HMAC タグ、同一のメッセージ、HMAC KMS キー、および 元の HMAC タグを計算するために GenerateMac が使用した MAC アルゴリズムを提供する必要があります。VerifyMac オペレーションは HMAC タグを計算し、それが提供された HMAC タグと同一であることを検証します。入力と計算された HMAC タグが同一ではない場合は、検証が失敗します。

HMAC KMS キーは [自動キーローテーション](#) をサポートしていないため、[カスタムキーストア](#) で HMAC KMS キーを作成することはできません。

AWS サービス内のデータを暗号化するための KMS キーを作成する場合は、対称暗号化キーを使用します。HMAC KMS キーを使用することはできません。

リージョン

HMAC KMS キーは、AWS KMS がサポート AWS リージョン するすべての でサポートされています。

詳細はこちら

- HMAC KMS キーを作成するには、「[HMAC KMS キーを作成する](#)」を参照してください。
- マルチリージョンの HMAC KMS キーを作成するには、「[のマルチリージョンキー AWS KMS](#)」を参照してください。
- AWS KMS コンソールが HMAC KMS キーに設定するデフォルトキーポリシーの違いを確認するには、「」を参照してください[the section called “暗号化オペレーションで KMS キーを使用することをキーユーザーに許可する”](#)。
- HMAC KMS キーの識別と表示については、「[HMAC KMS キーの特定](#)」を参照してください。
- JSON Web トークンを作成するための HMAC の使用について学ぶには、AWS セキュリティブログの「[How to protect HMACs inside AWS KMS](#)」(内で HMAC を保護する方法) を参照してください。
- ポッドキャストを聴く: AWS 公式ポッドキャストにて[AWS Key Management Serviceが HMAC を導入](#)。

の ML-DSA キー AWS KMS

AWS Key Management Service (AWS KMS) は、ポスト量子暗号化署名の Module-Lattice Digital Signature Algorithm (ML-DSA) をサポートしています。この実装は、将来の量子コンピューティングの脅威から保護するために、[連邦情報処理規格 \(FIPS\) 204 標準](#)に準拠しています。は、FIPS 140-3 セキュリティレベル 3 検証済みハードウェアセキュリティモジュールですべての ML-DSA キーと署名オペレーション AWS KMS を作成および保護します。セキュリティとパフォーマンスのバランスを取るために、の ML-DSA AWS KMS は、ML_DSA_44、ML_DSA_65、ML_DSA_87 の異なるキー仕様を通じて 3 つの異なるセキュリティレベルを提供します。

AWS KMS は、メッセージタイプを使用して最大 4 KB のRAWメッセージの非対称キー署名をサポートします。より大きなメッセージの場合、NIST FIPS 204 セクション 6.2 で定義されているように、ML-DSA 署名で使用される 64 バイトのメッセージ表現 μ を外部で計算する必要があります。AWS KMS [署名](#)オペレーションでEXTERNAL_MUメッセージタイプを使用して、この事前処理された 64 バイトのメッセージを指定します。外部で計算された μ によって生成される署名は、同じメッ

セージとプライベートキーを使用する場合の署名と同じRAWです。この署名は、NIST FIPS 204 のセクション 5.4 の「ハッシュ前」ML-DSA または HashML-DSA とは異なることに注意してください。

ML-DSA と EXTERNAL_MU メッセージタイプの使用の詳細については、「」を参照してください [ML-DSA キー仕様](#)。

ML-DSA と EXTERNAL_MU メッセージタイプを使用する例については、「」を参照してください [ML-DSA キーペアによるオフライン検証](#)。

のマルチリージョンキー AWS KMS

AWS KMS はマルチリージョンキーをサポートします。マルチリージョンキーは、複数のリージョン AWS KMS keys で同じキーを持っていたかのように AWS リージョン、同じ意味で使用できる異なる にあります。関連するマルチリージョンキーの各セットには同じキーマテリアルと [キー ID](#) があるため、再暗号化やクロスリージョン呼び出しを行う AWS リージョン ことなく、1 つの でデータを暗号化 AWS リージョン し、別の で復号できます AWS KMS。

すべての KMS キーと同様に、マルチリージョンキーが暗号化 AWS KMS されていないままになることはありません。暗号化または署名用の対称または非対称のマルチリージョンキーを作成する、HMAC タグの生成と検証用の HMAC マルチリージョンキーを作成する、および AWS KMS が生成するキーマテリアル、または [インポートされたキーマテリアルを持つマルチリージョンキー](#) を作成することができます。エイリアスおよびタグの作成、キーポリシーとグラントの設定、有効化/無効化の選択など、各マルチリージョンキーを個別に管理する必要があります。単一リージョンキーで実行できるすべての暗号化オペレーションで、マルチリージョンキーを使用できます。

マルチリージョンキーは、多くの一般的なデータセキュリティシナリオに対応する、柔軟で強力なソリューションです。

ディザスタリカバリ

バックアップおよびリカバリーキテクチャでは、マルチリージョンキーを使用すると、AWS リージョン 停止が発生した場合でも、暗号化されたデータを中断することなく処理できます。バックアップリージョンで保持されるデータはバックアップリージョンで復号し、バックアップリージョンで新たに暗号化されたデータは、そのリージョンの復元時にプライマリリージョンで復号することができます。

グローバルなデータ管理

グローバルに展開されるビジネスには、グローバルに配信され、AWS リージョン全体で一貫して利用可能なデータが必要です。データが存在するすべてのリージョンでマルチリージョンキー

を作成し、クロスリージョン呼び出しのレイテンシーや、各リージョンで異なるキーのデータの再暗号化に掛かるコストなしで、単一リージョンキーであるかのようにキーを使用できます。

配信署名アプリケーション

クロスリージョン署名機能を必要とするアプリケーションでは、マルチリージョンの非対称署名キーを使用して、異なる AWS リージョンで同一のデジタル署名を、一貫して繰り返し生成することができます。

単一のグローバルトラストストア (単一のルート認証機関 (CA))、およびルート CA によって署名されたリージョンの中間 CA で証明書チェーンを使用する場合、マルチリージョンキーは不要です。ただし、アプリケーション署名などの中間 CA がシステムでサポートされない場合は、マルチリージョンキーを使用して、リージョンの認定に一貫性を持たせることができます。

複数のリージョンにまたがるアクティブ-アクティブアプリケーション

一部のワークロードとアプリケーションは、アクティブ-アクティブアーキテクチャで複数のリージョンにまたがることができます。これらのアプリケーションでは、マルチリージョンキーを使用して、リージョンの境界を越えて移動する可能性のあるデータに対する暗号化と復号の同時オペレーションに同じキーマテリアルを提供し、複雑さを軽減できます。

マルチリージョンキーは、クライアント側の暗号化ライブラリ ([AWS Encryption SDK](#)、[AWS データベース暗号化 SDK](#)、[Amazon S3 クライアント側暗号化](#)など) で使用できます。

保管時の暗号化またはデジタル署名のために [AWS と統合する サービスは AWS KMS](#)、現在、マルチリージョンキーを単一リージョンキーであるかのように扱います。リージョン間で移動されたデータを再ラップまたは再暗号化する場合があります。例えば、Amazon S3 クロスリージョンレプリケーションでは、マルチリージョンキーで保護されたオブジェクトをレプリケートする場合でも、コピー先リージョンの KMS キーでデータを復号および再暗号化します。

マルチリージョンキーはグローバルではありません。マルチリージョンのプライマリキーを作成し、そのキーを [AWS パーティション](#) 内で選択するリージョンにレプリケートします。次に、各リージョンでマルチリージョンキーを個別に管理します。AWS も、ユーザーに代わってマルチリージョンキーを自動的に作成またはレプリケート AWS KMS することはありません。AWS は [AWS マネージドキー](#)、サービスがアカウントで作成する KMS キーであり、常に単一リージョンキーです。

中国リージョンでは、マルチリージョンキー機能を使用して、中国リージョンパーティション () 内に KMS キーをレプリケートできます `aws-cn`。たとえば、キーを中国 (北京) リージョンから中国 (寧夏) リージョンに、またはその逆にレプリケートできます。中国リージョン間でキーをレプリケートすることで、レプリケート先リージョン AWS Key Management Service のを使用することに同意

し、レプリケート先リージョンに適用されるすべての利用規約を遵守することになります。北京および寧夏リージョンから中国 AWS リージョンパーティション外のリージョンにキーをレプリケートすることはできません。同様に、中国リージョンパーティション外のリージョンから北京および寧夏リージョンにキーをレプリケートすることはできません。

既存の単一リージョンキーをマルチリージョンキーに変換することはできません。この設計により、既存の単一リージョンキーで保護されているすべてのデータが、同じデータ常駐プロパティとデータ主権プロパティを維持できます。

ほとんどのデータセキュリティニーズでは、リージョンリソースのリージョン分離と耐障害性により、標準の AWS KMS 単一リージョンキーが最適なソリューションになります。ただし、複数のリージョンにまたがるクライアント側のアプリケーションでデータを暗号化または署名する必要がある場合は、マルチリージョンキーがソリューションとなることがあります。

リージョン

マルチリージョンキーは、がサポート AWS リージョン する AWS KMS すべての でサポートされています。

料金とクォータ

関連するマルチリージョンキーのセットに含まれるすべてのキーは、料金およびクォータに関して、1 つの KMS キーとしてカウントされます。[AWS KMS クォータ](#)は、アカウントのリージョンごとに個別に計算されます。各リージョンのマルチリージョンキーの使用と管理は、そのリージョンのクォータでカウントされます。

サポートされる KMS キータイプ

次の種類のマルチリージョン KMS キーを作成できます。

- 対称暗号化 KMS キー
- 非対称 KMS キー
- HMAC KMS キー
- インポートされたキーマテリアルを持つ KMS キー

カスタムキーストアでマルチリージョンキーを作成することはできません。

詳細はこちら

- マルチリージョン KMS キーへのアクセスを制御する方法については、「[マルチリージョンキーへのアクセスを制御する](#)」を参照してください。
- 任意のタイプのマルチリージョンプライマリ KMS キーの作成方法については、「[マルチリージョンのプライマリキーを作成する](#)」を参照してください。
- マルチリージョンのレプリカ KMS キーの作成方法については、「[マルチリージョンのレプリカキーを作成する](#)」を参照してください。
- プライマリリージョンの更新方法については、「[マルチリージョンキーセットのプライマリキーを変更する](#)」を参照してください。
- マルチリージョン KMS キーの特定と表示については、「[HMAC KMS キーの特定](#)」を参照してください。
- マルチリージョン KMS キーの削除に関する注意事項については、「[Deleting multi-Region keys](#)」を参照してください。

用語と概念

マルチリージョンキーでは、次の条件と概念を使用します。

マルチリージョンキー

マルチリージョンキーは、異なる AWS リージョンで同じキー ID とキーマテリアル (およびその他の[共有プロパティ](#)) を持つ KMS キーのセットの 1 つです。各マルチリージョンキーは、完全に機能する KMS キーで、関連するマルチリージョンキーとは完全に独立して使用できます。関連するすべてのマルチリージョンキーは同じキー ID とキーマテリアルを持つため、相互運用可能です。つまり、任意の関連するマルチリージョンキーは、他の関連するマルチリージョンキーによって暗号化された暗号文を復号 AWS リージョン できます。

KMS キーの作成時に、KMS キーのマルチリージョンのプロパティを設定します。既存のキーでマルチリージョンプロパティを変更することはできません。単一リージョンキーをマルチリージョンキーに変換したり、マルチリージョンキーを単一リージョンキーに変換したりすることはできません。既存のワークロードをマルチリージョンシナリオに移動するには、データを再暗号化するか、新しいマルチリージョンキーを使用して新しい署名を作成する必要があります。

マルチリージョンキーは[対称または非対称](#)で、AWS KMS キーマテリアルまたは[インポートされたキーマテリアル](#)を使用できます。[カスタムキーストア](#)でマルチリージョンキーを作成することはできません。

関連するマルチリージョンキーのセットには、常に 1 つだけ[プライマリキー](#)があります。他の AWS リージョンで、そのプライマリキーの[レプリカキー](#)を作成できます。[プライマリリージョンを更新](#)

[する](#)と、プライマリキーがレプリカキーに変更され、指定されたレプリカキーがプライマリキーに変更されます。ただし、各キーに保持できるプライマリキーまたはレプリカキーは 1 つだけです AWS リージョン。リージョンはすべて、同じ [AWS パーティション](#) である必要があります。

関連するマルチリージョンキーの複数のセットを、同じまたは異なる AWS リージョンで持つことができます。関連するマルチリージョンキーは相互運用可能ですが、関連しないマルチリージョンキーは相互運用できません。

プライマリキー

マルチリージョンプライマリキーは、同じパーティション AWS リージョン 内の他の にレプリケートできる KMS キーです。マルチリージョンキーの各セットには、プライマリキーが 1 つしかありません。

プライマリキーは、次の点でレプリカキーとは異なります。

- プライマリキーのみが [レプリケーション](#) 可能です。
- プライマリキーは、[レプリカキー](#) (キーマテリアルとキー ID を含む) の [共有プロパティ](#) のソースです。
- [自動キーローテーション](#) は、プライマリキーでのみ有効または無効にできます。
- [プライマリキーの削除をいつでもスケジュールする](#) ことができます。ただし AWS KMS 、すべてのレプリカキーが削除されるまで、プライマリキーは削除されません。

プライマリキーとレプリカキーは、暗号化プロパティにおいて違いはありません。プライマリキーとそのレプリカキーは、同じ意味で 사용할 ことができます。

プライマリキーをレプリケートする必要はありません。プライマリキーは、KMS キーと同じように使用し、有用であればレプリケートすることができます。ただし、マルチリージョンキーには単一リージョンキーとは異なるセキュリティプロパティがあるため、プライマリキーをコレプリケートする場合にのみ、マルチリージョンキーを作成することをお勧めします。

レプリカキー

マルチリージョンのレプリカキーは、[プライマリキー](#) および関連するレプリカキーと同じ [キー ID](#) とキーマテリアルを持ちますが、異なる AWS リージョンに存在する KMS キーです。

レプリカキーは、固有のキーポリシー、グラント、エイリアス、タグ、およびその他のプロパティを持つ、完全に機能する KMS キーです。レプリカキーは、プライマリキーまたは他のキーのコピーまたはポイントではありません。プライマリキーと関連するすべてのレプリカキーが無効になっている

場合でも、レプリカキーを使用できます。また、レプリカキーをプライマリキーに変換し、プライマリキーをレプリカキーに変換することもできます。レプリカキーが作成されると、レプリカキーはそのプライマリキーに[キーローテーション](#)および[プライマリリージョンの更新](#)のみを依存します。

プライマリキーとレプリカキーは、暗号化プロパティにおいて違いはありません。プライマリキーとそのレプリカキーは、同じ意味で使うことができます。プライマリキーまたはレプリカキーで暗号化されたデータは、同じキー、または関連する任意のプライマリキーまたはレプリカキーで復号できます。

レプリケーション

マルチリージョンの[プライマリキー](#)を同じパーティション内の別の AWS リージョン にレプリケートできます。これを行うと、 はプライマリ[キーと同じキー ID と他の共有プロパティを使用して、指定されたリージョンにマルチリージョンレプリカキー](#) AWS KMS を作成します。 [??? ???](#)次に、キーマテリアルをリージョンの境界を越えて安全に転送し、すべて AWS KMS内で、新しいレプリカキーに関連付けます。

共有プロパティ

共有プロパティは、レプリカキーと共有されるマルチリージョンプライマリキーのプロパティです。は、プライマリキーと同じ共有プロパティ値を持つレプリカキー AWS KMS を作成します。次に、プライマリキーの共有プロパティ値をレプリカキーに定期的に同期します。レプリカキーでは、これらのプロパティを設定できません。

以下は、マルチリージョンキーの共有プロパティです。

- [キー ID](#) — ([キー ARN](#) の Region 要素が異なります)。
- [キーマテリアル](#)
- [キーマテリアルのオリジン](#)
- [キー仕様](#)および暗号化アルゴリズム
- [キーの用途](#)
- [自動キーローテーション](#) — 自動キーローテーションは、プライマリキーでのみ有効または無効にできます。新しいレプリカキーは、共有キーマテリアルのすべてのバージョンで作成されます。詳細については、「[Rotating multi-Region keys](#)」を参照してください。
- [オンデマンドローテーション](#) — プライマリキーでのみオンデマンドローテーションを実行できます。新しいレプリカキーは、共有キーマテリアルのすべてのバージョンで作成されます。詳細については、「[Rotating multi-Region keys](#)」を参照してください。

関連するマルチリージョンキーのプライマリおよびレプリカの指定は、共有プロパティと考えることもできます。[新しいレプリカキーを作成する](#)か、[プライマリキーを更新すると](#)、は関連するすべてのマルチリージョンキーに変更を AWS KMS 同期します。これらの変更が完了すると、関連するすべてのマルチリージョンキーが、プライマリキーとレプリカキーを正確に一覧表示します。

マルチリージョンキーのその他のプロパティはすべて、独立したプロパティです (説明、[キーポリシー](#)、[グラント](#)、[有効および無効キーステータス](#)、[エイリアス](#)、[タグ](#)を含む)。関連するすべてのマルチリージョンキーでこれらのプロパティに同じ値を設定できますが、独立したプロパティの値を変更すると、AWS KMS は同期しません。

マルチリージョンキーの共有プロパティの同期を追跡できます。AWS CloudTrail ログで、[SynchronizeMultiRegionKey](#) イベントを探します。

マルチリージョンキーのセキュリティに関する考慮事項

AWS KMS マルチリージョンキーは、必要な場合にのみ使用します。マルチリージョンキーは、暗号化されたデータを AWS リージョン の間で移動するワークロード、またはクロスリージョンアクセスが必要なワークロードに柔軟でスケーラブルなソリューションを提供します。保護されたデータを、リージョンを超えて共有、移動、バックアップする必要がある場合、または異なるリージョンで同一のデジタル署名を作成する必要がある場合は、マルチリージョンキーを検討します。

ただし、マルチリージョンキーを作成するプロセスでは、キーマテリアルは AWS KMS内の AWS リージョン の境界を越えて移動します。マルチリージョンキーによって生成される暗号文は、複数の地理的位置にある複数の関連キーによって復号される可能性があります。地域的に孤立したサービスやリソースにも大きな利点があります。各 AWS リージョン は他のリージョンから分離され、独立しています。リージョンでは耐障害性や安定性が提供され、レイテンシーを低減することもできます。これにより、別のリージョンの障害の影響を受けずに利用できる冗長リソースを作成できます。また AWS KMS、では、すべての暗号文を 1 つのキーのみで復号化できます。

マルチリージョンキーは、セキュリティに関する新しい考慮事項も提起します。

- マルチリージョンキーでは、アクセス制御とデータセキュリティポリシーの適用がより複雑です。複数の独立したリージョン全体で、ポリシーがキーで一貫して監査されることを確認する必要があります。また、個別のキーに依存する代わりに、ポリシーを使用して境界を適用する必要があります。

例えば、あるリージョンの給与チームが別のリージョンの給与データを読み取れないようにするために、データにポリシー条件を設定する必要があります。また、あるリージョンのマルチリージョンキーが 1 つのテナントのデータを保護し、別のリージョンの関連するマルチリージョンキーが

別のテナントのデータを保護するというシナリオを防ぐには、アクセス制御を使用する必要があります。

- リージョン全体のキーの監査もより複雑です。マルチリージョンキーでは、複数のリージョン全体の監査アクティビティを調べて調整し、保護されるデータのキーアクティビティを完全に理解する必要があります。
- データ常駐に関するコンプライアンスは、より複雑になる可能性があります。孤立したリージョンでは、データ常駐とデータ主権のコンプライアンスを確保できます。特定のリージョンの KMS キーは、そのリージョン内の機密データのみを復号できます。あるリージョンで暗号化されたデータは完全に保護され、他のリージョンではアクセスできません。

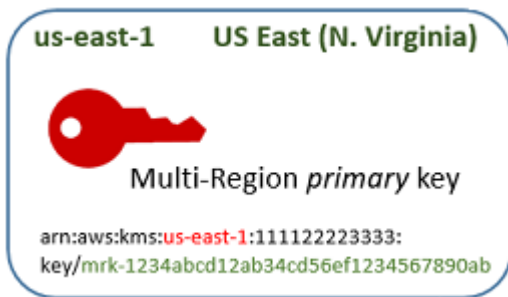
マルチリージョンキーを使用してデータレジデンシーとデータ主権を検証するには、アクセスポリシーを実装し、複数のリージョンにまたがって AWS CloudTrail イベントをコンパイルする必要があります。

マルチリージョンキーのアクセス制御を管理しやすくするために、マルチリージョンキーのレプリケーション許可 ([kms:ReplicateKey](#)) は、キーを作成するスタンダードなアクセス許可 ([kms:CreateKey](#)) と分けられています。また、は、マルチリージョンキーを作成 `kms:MultiRegion`、使用、または管理するアクセス許可を許可または拒否する や、マルチリージョンキーをレプリケートできるリージョンを制限する など `kms:ReplicaRegion`、マルチリージョンキーの複数のポリシー条件 AWS KMS をサポートします。詳細については、「[マルチリージョンキーへのアクセスを制御する](#)」を参照してください。

マルチリージョンキーの仕組み

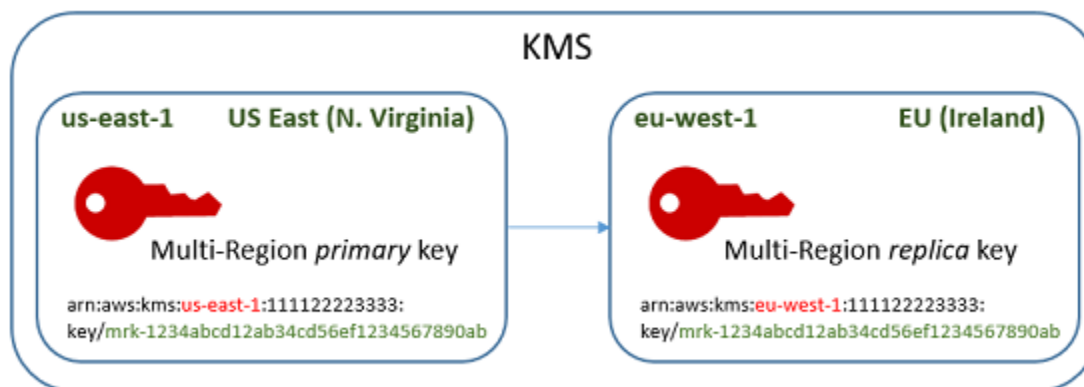
まず、米国東部 (バージニア北部) など、 が AWS KMS サポート AWS リージョン する で対称または非対称 [マルチリージョンのプライマリキー](#) を作成します。キーを単一リージョンにするか、マルチリージョンにするかは、作成時にのみ決定できます。このプロパティは後で変更できません。KMS キーと同様に、マルチリージョンキーのキーポリシーを設定することで、グラントを作成したり、分類と認可用のエイリアスとタグを追加したりできます。(これらは、他のキーと共有または同期されない [独立したプロパティ](#) です)。暗号化または署名の暗号化オペレーションで、マルチリージョンのプライマリキーを使用できます。

[マルチリージョンのプライマリキーは、コンソールで作成することも、パラメータを に設定して CreateKey API を使用して作成](#) することもできます `true`。AWS KMS [CreateKey](#) `MultiRegion` マルチリージョンキーには `mrk-` で始まる固有のキー ID があります。`mrk-` プレフィックスを使用して、プログラムで MRK を識別できます。



選択すると、マルチリージョンのプライマリキーを、欧州 (アイルランド) など、同じ[AWS パーティション](#) AWS リージョン 内の 1 つ以上の異なる に[レプリケート](#)できます。これを行うと、はプライマリキーと同じキー ID と他の共有プロパティを使用して、指定されたリージョンにレプリカキー AWS KMS を作成します。???次に、キーマテリアルをリージョンの境界を越えて安全に転送し、すべて AWS KMS内で、コピー先リージョンの新しい KMS キーに関連付けます。結果として、2 つの関連するマルチリージョンキーが作成されます (プライマリキーとレプリカキー)。これらは相互に使用することができます。

[マルチリージョンレプリカキーは、コンソールまたは ReplicateKey API を使用して作成](#)できます。
AWS KMS [ReplicateKey](#)



作成された[マルチリージョンレプリカキー](#)は、完全に機能する KMS キーで、プライマリキーと同じ[共有プロパティ](#)を備えています。それ以外の点では、独自の説明、キーポリシー、グラント、エイリアス、タグを持つ独立した KMS キーです。マルチリージョンキーを有効または無効にしても、関連するマルチリージョンキーには影響しません。プライマリキーとレプリカキーは、暗号化オペレーションで個別に使用することも、連携させて使用することもできます。例えば、米国東部 (バージニア北部) リージョンのプライマリキーを使用してデータを暗号化し、欧州 (アイルランド) リージョンにデータを移動し、レプリカキーを使用してデータを復号できます。

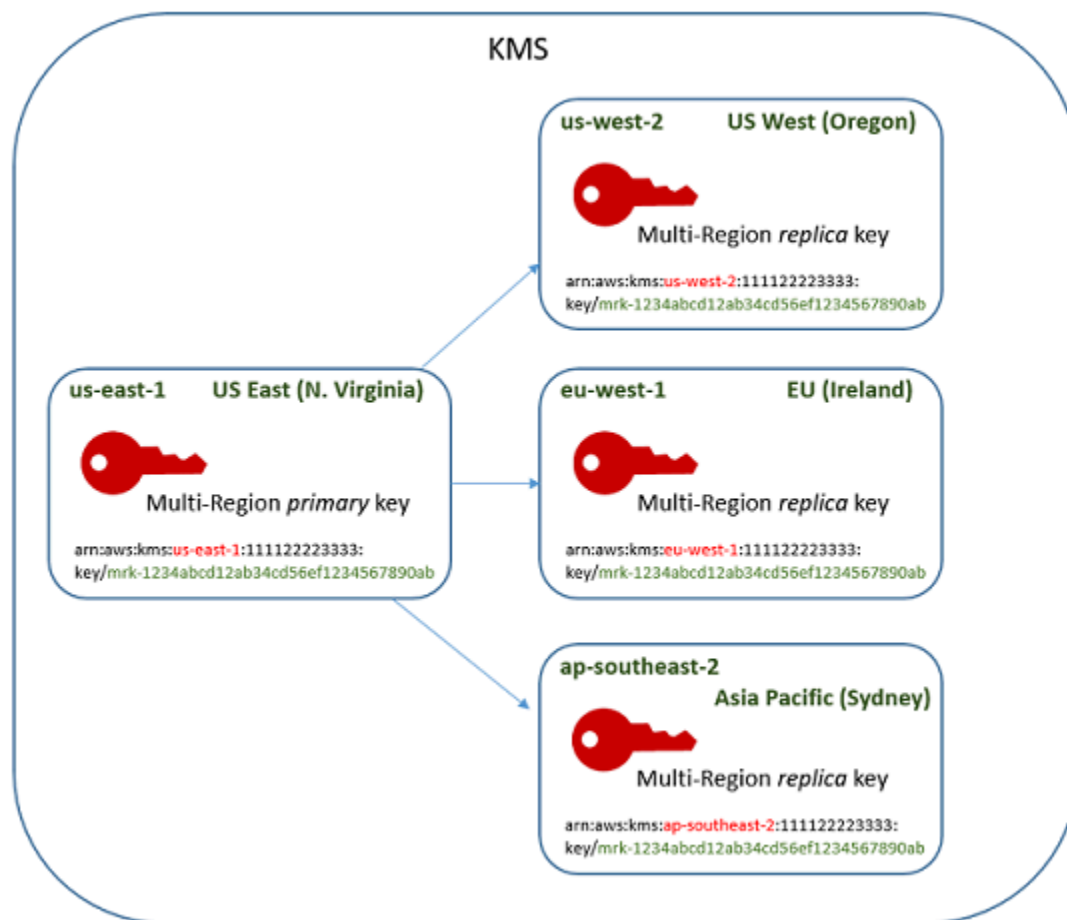
関連するマルチリージョンキーは同じキー ID を持ちます。キー ARN (Amazon リソースネーム) は、リージョンフィールドでのみ異なります。例えば、マルチリージョンのプライマリキーとレプリ

カキーには、次のキー ARN があります。キー ID (キー ARN の最後の要素) は同一です。両方のキーには、mrk- で始まる、マルチリージョンキー固有のキー ID があります。

```
Primary key: arn:aws:kms:us-east-1:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab
Replica key: arn:aws:kms:eu-west-1:111122223333:key/mrk-1234abcd12ab34cd56ef12345678990ab
```

相互運用性のために、同じキー ID が必要です。暗号化時に、は KMS キーのキー ID を暗号文に AWS KMS バインドするため、暗号文は、その KMS キーまたは同じキー ID を持つ KMS キーでのみ復号できます。この機能により、関連するマルチリージョンキーが認識しやすくなり、相互に使用しやすくなります。例えば、アプリケーションで使用する場合は、共有キー ID で関連するマルチリージョンキーを参照できます。次に、必要に応じて、リージョンまたは ARN を指定して、それらを区別します。

データのニーズの変化に応じて、米国西部 (オレゴン) やアジアパシフィック (シドニー) など、同じパーティション AWS リージョン 内の他の にプライマリーキーをレプリケートできます。結果は、次の図表に示された、同じキーマテリアルとキー ID を持つ 4 つの関連するマルチリージョンキーとなります。キーは個別に管理します。キーは独立して使用することも、連携させて使用することもできます。例えば、アジアパシフィック (シドニー) でレプリカキーを使用してデータを暗号化し、データを米国西部 (オレゴン) に移動して、米国西部 (オレゴン) でレプリカキーを使用して復号できます。



マルチリージョンキーに関するその他の考慮事項は次のとおりです。

共有プロパティの同期 — マルチリージョンキーの共有プロパティが変更された場合、はプライマリキーからすべてのレプリカキーに変更 AWS KMS を自動的に同期します。共有プロパティの同期をリクエストまたは強制することはできません。はすべての変更 AWS KMS を検出して同期します。ただし、CloudTrail ログの [SynchronizeMultiRegionKey](#) イベントを使用して、同期を監査することができます。

たとえば、対称マルチリージョンプライマリキーで自動キーローテーションを有効にすると、はその設定をすべてのレプリカキー AWS KMS にコピーします。キーマテリアルをローテーションすると、関連するすべてのマルチリージョンキー間でローテーションが同期されます。これにより、マルチリージョンキーは引き続き現在と同じキーマテリアルを持ち、古いバージョンのキーマテリアルすべてにアクセスできます。新しいレプリカキーを作成すると、そのキーマテリアルは、関連するすべてのマルチリージョンキーの現在のキーマテリアルと同じになり、以前のバージョンのキーマテリアルすべてにアクセスできます。詳細については、「[Rotating multi-Region keys](#)」を参照してください。

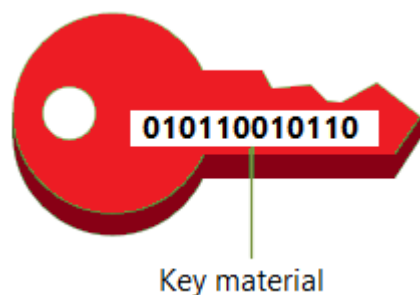
プライマリキーの変更 — マルチリージョンキーのすべてのセットには、プライマリキーが 1 つだけ必要です。[プライマリキー](#)はレプリケートできる唯一のキーです。また、レプリカキーの共有プロパティのソースでもあります。ただし、プライマリキーをレプリカに変更し、レプリカキーの 1 つをプライマリに昇格させることができます。これにより、特定のリージョンから複数リージョンのプライマリキーを削除したり、プロジェクト管理者の近くにあるリージョンでプライマリキーを検索したりできます。詳細については、「[マルチリージョンキーセットのプライマリキーを変更する](#)」を参照してください。

マルチリージョンキーの削除 — すべての KMS キーと同様に、マルチリージョンキーを削除する前に AWS KMS 削除をスケジュールする必要があります。キーが削除保留中の間は、暗号化オペレーションでキーを使用することはできません。ただし、すべてのレプリカキーが削除されるまで、マルチリージョンのプライマリキーを削除 AWS KMS しません。詳細については、[Deleting multi-Region keys](#) を参照してください

キーの AWS KMS キーマテリアルのインポート

指定したキーマテリアルを使用して AWS KMS keys (KMS キー) を作成できます。

KMS キーは、データキーの論理表現です。KMS キーのメタデータには、暗号化オペレーションの実行に使用されるキーマテリアルの ID が含まれます。[KMS キーを作成すると](#)、デフォルトではその KMS キーのキーマテリアル AWS KMS を生成します。ただし、キーマテリアルを使用せずに KMS キーを作成し、独自のキーマテリアルをその KMS キーにインポートできます。この機能は、「キーの持ち込み」(BYOK) とも呼ばれます。



Note

AWS KMS は、暗号化テキストがインポートされたキーマテリアルを持つ AWS KMS KMS キーで暗号化されていても AWS KMS、 の外部で対称暗号化 KMS キーによって暗号化された暗号化テキストの復号をサポートしていません。AWS KMS はこのタスクに必要な暗号化テキスト形式を発行せず、形式は予告なしに変更される可能性があります。

インポートされたキーマテリアルを使用する場合、[がキーマテリアルのコピーを使用](#) AWS KMS できるようにしながら、キーマテリアルの責任を引き続き負います。そうする場合の理由として次のものが考えられます (複数が組み合わさる場合もあります)。

- 要件を満たすエントロピーのソースを使用してキーマテリアルが生成されたことを証明するため。
- AWS サービスで独自のインフラストラクチャのキーマテリアルを使用し、[を使用して AWS KMS](#) そのキーマテリアルのライフサイクルを管理するには AWS。
- コード署名、PKI 証明書署名 AWS KMS、証明書固定アプリケーションのキーなど、既存の確立されたキーを [で使用する](#)には
- AWS でキーマテリアルの有効期限を設定し、[手動で削除します](#)が、今後再び使用できるようにするには。これに対して、[キー削除のスケジュール](#)は、7 日から 30 日間の待機時間が必要となり、その後、削除された KMS キーは復元できません。
- キーマテリアルの元のコピーを所有し、キーマテリアルの完全なライフサイクル中に耐久性とディザスタリカバリを強化するために [の外部 AWS](#) に保持します。
- 非対称キーと HMAC キーの場合、インポートすると、互換性があり相互運用可能なキーが作成されます。このキーは、内外で動作します AWS。

サポートされる KMS キータイプ

AWS KMS は、次のタイプの KMS キーのインポートされたキーマテリアルをサポートします。[カスタムキーストア](#)内の KMS キーにキーマテリアルをインポートすることはできません。

- [対称暗号化 KMS キー](#)
- [非対称 KMS キー \(ML-DSA キーを除く\)](#)
- [HMAC KMS キー](#)
- サポートされているすべてのタイプの[マルチリージョンキー](#)。

リージョン

インポートされたキーマテリアルは、[がサポート](#) AWS リージョン [する AWS KMS](#) すべての [でサポート](#)されています。

中国リージョンでは、対称暗号化 KMS キーの主要なマテリアル要件は他のリージョンとは異なります。詳細については、「[ステップ 3: キーマテリアルを暗号化する](#)」を参照してください。

詳細はこちら

- インポートされたキーマテリアルを持つ KMS キーの作成については、「[インポートされたキーマテリアルを持つ KMS キーの作成](#)」を参照してください。
- インポートされたキーマテリアルを持つ KMS キーの有効期限が近づいたときに通知するアラームの作成については、「[インポートされたキーマテリアルの有効期限を通知する CloudWatch アラームを作成する](#)」を参照してください。
- KMS キーにキーマテリアルを再インポートするには、「[キーマテリアルの再インポート](#)」を参照してください。
- オンデマンドローテーション用に新しいキーマテリアルを KMS キーにインポートするには、[新しいキーマテリアルをインポートする](#)「」および「」を参照してください。[オンデマンドキーローテーションの実行](#)。
- インポートされたキーマテリアルを持つ KMS キーの識別と表示については、「[インポートされたキーマテリアルを持つ KMS キーを特定する](#)」を参照してください。
- インポートされたキーマテリアルを持つ KMS キーの削除に関する注意事項については、「[Deleting KMS keys with imported key material](#)」を参照してください。

インポートされたキーマテリアルに関する注意事項

キーマテリアルを にインポートする前に AWS KMS、インポートされたキーマテリアルの次の特性を理解しておく必要があります。

キーのマテリアルを生成する

お客様の責任において、セキュリティ要件を満たすランダムソースを使用して、キーマテリアルを生成する必要があります。

可用性と耐久性に責任があります

AWS KMS は、インポートされたキーマテリアルを高可用性に保つように設計されています。ただし、AWS KMS は、インポートされたキーマテリアルの耐久性を、 が AWS KMS 生成するキーマテリアルと同じレベルで維持しません。詳細については、「[インポートされたキーマテリアルの保護](#)」を参照してください。

キーマテリアルを削除できます。

KMS キーから、[インポートしたキーマテリアルを削除](#)すると、KMS キーはただちに使用できなくなります。また、KMS キーにキーマテリアルをインポートするときに、キーの有効期間を確認して[有効期限を設定](#)することができます。有効期限が来ると、AWS KMS [はキーマテリアルを削除します](#)。キーマテリアルがない場合、KMS キーは暗号化オペレーションで使用することはできません。キーを復元するには、キーに同じキーマテリアルを再インポートする必要があります。

非対称キー、HMAC キー、マルチリージョンキーのキーマテリアルを変更することはできません

KMS キーにキーマテリアルをインポートすると、KMS キーはキーマテリアルに永続的に関連付けられます。[同じキーマテリアルを再インポート](#)することはできますが、別のキーマテリアルをその KMS キーにインポートすることはできません。また、キーマテリアルがインポートされた KMS キーに対して、[自動キーローテーションを有効にする](#)ことはできません。ただし、キーマテリアルがインポートされた [KMS キーを手動でローテーションする](#)ことはできます。

単一リージョンの対称暗号化キーでオンデマンドローテーションを実行できます。

インポートされたキーマテリアルを持つ単一リージョンの対称暗号化キーは、オンデマンドローテーションをサポートします。これらの[キーに複数のキーマテリアルをインポート](#)し、[オンデマンドローテーション](#)を使用して現在のキーマテリアルを更新できます。現在のキーマテリアルは暗号化と復号の両方に使用されますが、他の (最新ではない) キーマテリアルは復号にのみ使用できます。

キーマテリアルのオリジンは変更できません

インポートされたキーマテリアルのために設計されている KMS キーには、変更することができない EXTERNAL の[オリジン](#)の値があります。インポートされたキーマテリアルの KMS キーを変換して、他のソースからのキーマテリアルを使用することはできません AWS KMS。同様に、AWS KMS キーマテリアルを含む KMS キーを、インポートされたキーマテリアル用に設計されたキーマテリアルに変換することはできません。

キーマテリアルはエクスポートできません

インポートしたキーマテリアルをエクスポートすることはできません。インポートしたキーマテリアルをどのような形式でも返す AWS KMS ことはできません。インポートしたキーマテリアルのコピーを AWS の外部、できればハードウェアセキュリティモジュール (HSM) などのキーマネージャーに保持する必要があります。これにより、キーマテリアルを削除した場合や有効期限が切れた場合に再インポートできます。

インポートしたキーマテリアルを含むマルチリージョンキーを作成できます

インポートされたキーマテリアルを含むマルチリージョンには、インポートされたキーマテリアルの KMS キーと同様の機能があり、AWS リージョン間の相互運用が可能です。キーマテリアルがインポートされたマルチリージョンキーを作成するには、同じキーマテリアルをプライマリ KMS キーと各レプリカキーにインポートする必要があります。マルチリージョン対称暗号化キーは、オンデマンドローテーションをサポートしていません。

非対称キーと HMAC キーは移植可能で相互運用可能です

の外部で非対称キーマテリアルと HMAC キーマテリアルを使用して AWS、同じインポートされた AWS KMS キーマテリアルを持つキーと相互運用できます。

アルゴリズムで使用される AWS KMS KMS キーに厳密にバインドされている対称暗号文とは異なり、は暗号化、署名、MAC 生成に標準の HMAC 形式と非対称形式 AWS KMS を使用します。そのため、キーは移植可能で、従来のエスクローキーシナリオにも対応しています。

KMS キーにインポートされたキーマテリアルがある場合、の外部 AWS でインポートされたキーマテリアルを使用して、次のオペレーションを実行できます。

- HMAC キー — キーマテリアルがインポートされた HMAC KMS キーによって生成された HMAC タグを検証できます。インポートされたキーマテリアルで HMAC KMS キーを使用して、外のキーマテリアルによって生成された HMAC タグを検証することもできます AWS。
- 非対称暗号化キー — の外部でプライベート非対称暗号化キーを使用して AWS、対応するパブリックキーで KMS キーによって暗号化された暗号文を復号できます。非対称 KMS キーを使用して、の外部で生成された非対称暗号文を復号することもできます AWS。
- 非対称署名キー — インポートされたキーマテリアルで非対称署名 KMS キーを使用して、外部のプライベート署名キーによって生成されたデジタル署名を検証できます AWS。の外部で非対称パブリック署名キーを使用して AWS、非対称 KMS キーによって生成された署名を検証することもできます。
- 非対称キーアグリーメントキー — インポートされたキーマテリアルを持つ非対称キーアグリーメント KMS キーを使用して、AWSの外部のピアとの共有シークレットを取得できます。

同じキーマテリアルを同じ AWS リージョン内の別の KMS キーにインポートすると、それらのキーも相互運用できます。異なる で相互運用可能な KMS キーを作成するには AWS リージョン、インポートされたキーマテリアルを持つマルチリージョンキーを作成します。

対称暗号化キーは移植も相互運用もできません

AWS KMS が生成する対称暗号文は、移植可能または相互運用可能ではありません。AWS KMS は、移植に必要な対称暗号文形式を発行せず、形式は予告なしに変更される可能性があります。

- AWS KMS は、インポートしたキーマテリアルを使用しているても AWS、外部で暗号化した対称暗号文を復号できません。
- AWS KMS は、暗号化テキストがインポートされたキーマテリアルを持つ KMS キーで暗号化された場合でも AWS KMS、の外部での AWS KMS 対称暗号化テキストの復号をサポートしていません。

- インポートされた同じキーマテリアルの KMS キーは相互運用できません。各 KMS キーに固有の暗号文 AWS KMS を生成する対称暗号文。この暗号文形式により、データを暗号化した KMS キーのみが復号できることが保証されます。

また、[AWS Encryption SDK](#)や [Amazon S3 クライアント側の暗号化](#)などの AWS ツールを使用して対称 AWS KMS 暗号文を復号することはできません。

その結果、インポートされたキーマテリアルでキーを使用して、キーマテリアルへの条件付きアクセスを持つ許可されたサードパーティーが の外部で特定の暗号文を復号できるキーエスクローの配置をサポートすることはできません AWS KMS。キーエスクローをサポートするには、[AWS Encryption SDK](#) を使用して、AWS KMSに依存しないキーでメッセージを暗号化します。

インポートされたキーマテリアルの保護

インポートしたキーマテリアルは、転送中も保管中も保護されます。キーマテリアルをインポートする前に、[FIPS 140-3 暗号化モジュール検証プログラム](#)で検証された AWS KMS ハードウェアセキュリティモジュール (HSMs) を使します。キーマテリアルをラッピングパブリックキーで直接暗号化することも、キーマテリアルを AES 対称キーで暗号化してから、AES 対称キーを RSA パブリックキーで暗号化することもできます。

受信すると、 は AWS KMS HSM 内の対応するプライベートキーを使用してキーマテリアルを AWS KMS 復号し、HSM の揮発性メモリにのみ存在する AES 対称キーで再暗号化します。キーマテリアルがプレーンテキストで HSM 外に出ることはありません。使用中のみ、AWS KMS HSMs。

キーマテリアルがインポートされた KMS キーの使用は、KMS キーに設定した[アクセス制御ポリシー](#)によってのみ特定されます。さらに、[エイリアス](#)と[タグ](#)を使用して、KMS キーを識別し、KMS キーへの[アクセスを制御できます](#)。キーを[有効化または無効化](#)したり、[表示](#)したり、あるいは AWS CloudTrailのようなサービスを使用して[モニタリング](#)したりすることができます。

ただし、キーマテリアルのフェイルセーフコピーはお客様のみが管理します。この追加のコントロール指標の見返りとして、インポートされたキーマテリアルの耐久性と全体的な可用性に責任があります。AWS KMS は、インポートされたキーマテリアルを高可用性に保つように設計されています。ただし、AWS KMS が AWS KMS 生成するキーマテリアルと同じレベルで、インポートされたキーマテリアルの耐久性は維持されません。

耐久性におけるこの相違は、次の場合に有意義です。

- インポートしたキーマテリアル[の有効期限を設定すると](#)、 は有効期限が切れた後にキーマテリアル AWS KMS を削除します。AWS KMS は KMS キーまたはそのメタデータを削除しません。イ

ンポートされたキーマテリアルの有効期限が近づいたときに通知する [Amazon CloudWatch アラームを作成できます](#)。

が KMS キーに対して AWS KMS 生成するキーマテリアルを削除したり、AWS KMS キーマテリアルの有効期限を設定したりすることはできません。

- [インポートされたキーマテリアルを手動で削除する](#)と、はキーマテリアル AWS KMS を削除しますが、KMS キーまたはそのメタデータは削除しません。対照的に、[キーの削除をスケジュール](#)するには、7～30 日間の待機期間が必要です。その後、は KMS キー、そのメタデータ、およびそのキーマテリアル AWS KMS を完全に削除します。
- 万一、に影響するリージョン全体の特定の障害 AWS KMS (停電など) が発生した場合、AWS KMS はインポートされたキーマテリアルを自動的に復元できません。ただし、は KMS キーとそのメタデータを復元 AWS KMS できます。

インポートされたキーマテリアルのコピーは、ユーザーが管理するシステムの の外部 AWS に保持する必要があります。インポートされたキーマテリアルのエクスポート可能なコピーを、HSM などのキーマネジメントシステムに保存しておくことをお勧めします。ベストプラクティスとして、KMS キー ARN と によって生成されたキーマテリアル ID への参照を、キーマテリアルのエクスポート可能なコピー AWS KMS とともに保存する必要があります。インポートしたキーマテリアルが削除されるか期限切れになった場合、同じキーマテリアルを再インポートするまで、関連付けられた KMS キーは使用できなくなります。インポートしたキーマテリアルが完全に失われた場合、KMS キーで暗号化された暗号文は回復できません。

Important

単一リージョンの対称暗号化キーには、複数のキーマテリアルを関連付けることができます。これらのキーマテリアルのいずれかを削除するか、それらのキーマテリアルのいずれかの有効期限が切れるとすぐに、KMS キー全体が使用できなくなります (削除または期限切れのキーマテリアルがでない限り PENDING_ROTATION)。キーが暗号化オペレーションで使用できるようになる前に、そのようなキーに関連付けられた期限切れまたは削除されたキーマテリアルを再インポートする必要があります。

CloudHSM キーストアの KMS キー

AWS CloudHSM キーストア AWS KMS keys で の作成、表示、管理、使用、削除のスケジュールを設定できます。手順は、他の KMS キーを使用する際の手順に非常によく似ています。唯一の違いは、KMS AWS CloudHSM キーを作成するときに キーストアを指定することです。次に、は、

キーストアに関連付けられているクラスター内の AWS CloudHSM KMS キーの抽出不可能な AWS CloudHSM キーマテリアル AWS KMS を作成します。キーストアで KMS AWS CloudHSM キーを使用すると、[暗号化オペレーション](#)はクラスター内の HSMs で実行されます。

サポートされている機能

このセクションで説明する手順に加えて、キーストアの KMS AWS CloudHSM キーを使用して以下を実行できます。

- KMS キーへの[アクセスを承認する](#)ときは、キーポリシー、IAM ポリシー、グラントを使用します。
- KMS キーを[有効および無効にします](#)。
- [タグ](#)を割り当てて[エイリアス](#)を作成し、属性ベースのアクセス制御 (ABAC) を使用して KMS キーへのアクセスを承認します。
- KMS キーを使用して以下の暗号化オペレーションを実行します。
 - [暗号化](#)
 - [Decrypt](#)
 - [GenerateDataKey](#)
 - [GenerateDataKeyWithoutPlaintext](#)
 - [ReEncrypt](#)

非対称データキーペアを生成するオペレーション、[GenerateDataKeyPair](#) と [GenerateDataKeyPairWithoutPlaintext](#) は、カスタムキーストアでサポートされていません。

- [AWS KMSを統合し、カスタマーマネージドキーをサポートするAWS サービス](#)で KMS キーを使用します。
- [AWS CloudTrail ログ](#)と [Amazon CloudWatch モニタリングツール](#)での KMS キーの使用状況を追跡します。

サポートされていない 機能

- AWS CloudHSM キーストアは、対称暗号化 KMS キーのみをサポートします。キーストアで HMAC KMS キー、非対称 KMS キー、または非対称データ AWS CloudHSM キーペアを作成することはできません。
- [キーストアの KMS キーにキーマテリアルをインポート](#)することはできません。は、クラスター内の AWS CloudHSM KMS キーのキーマテリアル AWS KMS を生成します。AWS CloudHSM
- キーストアの KMS キーの AWS CloudHSM キーマテリアル[の自動ローテーション](#)を有効または無効にすることはできません。

キーストアでの KMS AWS CloudHSM キーの使用

リクエストで KMS キーを使用する場合は、ID またはエイリアスで KMS キーを識別します。AWS CloudHSM キーストアまたは AWS CloudHSM クラスターを指定する必要はありません。レスポンスには、対称暗号化 KMS キーについて返されるものと同じフィールドが含まれます。

ただし、キーストアで KMS AWS CloudHSM キーを使用する場合、暗号化オペレーションは AWS CloudHSM キーストアに関連付けられた AWS CloudHSM クラスター内で完全に実行されます。オペレーションでは、選択した KMS キーに関連付けられているクラスターのキーマテリアルが使用されます。

これを可能にするには、次の条件が必要です。

- KMS キーの[キーストア](#)は Enabled である必要があります。キーステータスを検索するときは、[AWS KMS コンソール](#)の [Status] (ステータス) フィールド、または [DescribeKey](#) レスポンスの KeyState フィールドを使用します。
- AWS CloudHSM キーストアは AWS CloudHSM クラスターに接続されている必要があります。[AWS KMS コンソール](#)の [Status] (ステータス) または [DescribeCustomKeyStores](#) レスポンスの ConnectionState は、CONNECTED になっているはずです。
- カスタムキーストアに関連付けられている AWS CloudHSM クラスターには、少なくとも 1 つのアクティブな HSM が含まれている必要があります。クラスター内のアクティブな HSMs の数を確認するには、[AWS KMS コンソール](#)、AWS CloudHSM コンソール、または [DescribeClusters](#) オペレーションを使用します。
- AWS CloudHSM クラスターには、KMS キーのキーマテリアルが含まれている必要があります。キーマテリアルがクラスターから削除された場合、または HSM がキーマテリアルを含まないバックアップから作成された場合、暗号化オペレーションは失敗します。

これらの条件が満たされない場合、暗号化オペレーションは失敗し、は KMSInvalidStateException 例外 AWS KMS を返します。通常、[AWS CloudHSM キーストアを再接続するだけで済みます](#)。その他のヘルプについては、「[失敗した KMS キーを修正するには](#)」を参照してください。

キーストアで KMS AWS CloudHSM キーを使用する場合は、各 AWS CloudHSM キーストアの KMS キーが[暗号化オペレーションのカスタムキーストアリクエストクォータ](#)を共有することに注意してください。クォータを超えると、は AWS KMS を返します ThrottlingException。AWS CloudHSM キーストアに関連付けられている AWS CloudHSM クラスターが、AWS CloudHSM キーストアに関連しないコマンドなど、多数のコマンドを処理している場合、のレート ThrottlingException がさらに低くなる可能性があります。すべてのリクエストで

ThrottlingException が表示される場合、リクエスト速度を下げ、再度コマンドを試してください。カスタムキーストアのリクエストクォータの詳細については、「[カスタムキーストアのリクエストクォータ](#)」を参照してください。

詳細

- AWS CloudHSM キーストアの詳細については、「」を参照してください[AWS CloudHSM キーストア](#)。
- キーストアで KMS AWS CloudHSM キーを作成するには、「」を参照してください[キーストアに KMS AWS CloudHSM キーを作成する](#)。
- キーストアで KMS AWS CloudHSM キーを識別して表示するには、「」を参照してください[キーストア内の KMS AWS CloudHSM キーを特定する](#)。
- キーストアで KMS キーと AWS CloudHSM キーマテリアルを検索するには、「」を参照してください[キーストアで KMS キーと AWS CloudHSM キーマテリアルを検索する](#)。
- AWS CloudHSM キーストアで KMS キーを削除するための特別な考慮事項については、「[キーストアから KMS AWS CloudHSM キーを削除する](#)」を参照してください。

外部キーストアの KMS キー

外部キーストアで KMS キーを作成、表示、管理、使用し、削除をスケジュールするには、他の KMS キーに使用する手順と極めてよく似た手順を使用します。ただし、外部キーストアに KMS キーを作成する場合は、[外部キーストア](#)と[外部キー](#)を指定します。外部キーストアで KMS キーを使用すると、指定された外部キーにより、外部キーマネージャーが[暗号化および復号オペレーション](#)を実行します。

AWS KMS は、外部キーマネージャーで暗号化キーを作成、表示、更新、または削除することはできません。は、外部キーマネージャーまたは外部キーに直接アクセス AWS KMS することはありません。暗号化オペレーションのリクエストはすべて、[外部キーストアプロキシ](#)によって仲介されます。外部キーストアで KMS キーを使用するには、KMS キーをホストする外部キーストアを、外部キーストアプロキシに[接続](#)する必要があります。

サポートされている機能

このセクションで説明する手順に加えて、外部キーストアでは KMS キーを使用して次のことを実行できます。

- [キーポリシー](#)、[IAM ポリシー](#)、[グラント](#)を使用して、KMS キーへのアクセスを管理します。
- KMS キーを[有効および無効](#)にします。これらのアクションは、外部キーマネージャーの外部キーには影響しません。

- [タグ](#)を割り当てて[エイリアス](#)を作成し、[属性ベースのアクセス制御](#) (ABAC) を使用して KMS キーへのアクセスを承認します。
- KMS キーを使用して以下の暗号化オペレーションを実行します。
 - [暗号化](#)
 - [Decrypt](#)
 - [GenerateDataKey](#)
 - [GenerateDataKeyWithoutPlaintext](#)
 - [ReEncrypt](#)

非対称データキーペアを生成するオペレーション、[GenerateDataKeyPair](#) と [GenerateDataKeyPairWithoutPlaintext](#) は、カスタムキーストアでサポートされていません。

- [AWS KMSを統合し](#)、[カスタマーマネージドキー](#)をサポートするAWS のサービス で KMS キーを使用します。

サポートされていない 機能

- 外部キーストアは、[対称暗号化 KMS キー](#)のみをサポートしています。外部キーストアで HMAC KMS キーや非対称 KMS キーを作成することはできません。
- 外部キーストアの KMS キーでは、[GenerateDataKeyPair](#) および [GenerateDataKeyPairWithoutPlaintext](#) はサポートされていません。
- [AWS::KMS::Key AWS CloudFormation テンプレート](#)を、外部キーストアの作成や外部キーストア内の KMS キー作成に使用することはできません。
- [マルチリージョンキー](#)は、外部キーストアではサポートされていません。
- [キーマテリアルがインポートされた](#) KMS キーは、外部キーストアではサポートされていません。
- [自動キーローテーション](#)は、カスタムキーストアの KMS キーではサポートされていません。

外部キーストアで KMS キーを使用する

リクエストで KMS キーを使用すると、[キー ID](#)、[キー ARN](#)、[エイリアス](#)、[エイリアス ARN](#) により KMS キーを識別します。外部キーストアを指定する必要はありません。レスポンスには、対称暗号化 KMS キーについて返されるものと同じフィールドが含まれます。ただし、外部キーストアで KMS キーを使用すると、暗号化および複合化オペレーションは KMS キーに関連付けられた外部キーを使用して、外部キーマネージャーによって実行されます。

外部キーストアの KMS キーによって暗号化された暗号文が、標準 KMS キーによって暗号化された暗号文と少なくとも同じ安全性を確保するために、は [二重暗号化](#) AWS KMS を使用しま

す。データは、まず AWS KMS キーマテリアル AWS KMS を使用して で暗号化されます。次に、KMS キーの外部キーを使用して、外部キーマネージャーによって暗号化されます。二重に暗号化された暗号文を復号するには、まず KMS キーの外部キーを使用して、外部キーマネージャーによって暗号文を復号します。次に、KMS AWS KMS キーのキーマテリアル AWS KMS を使用して で復号されます。

これを可能にするには、次の条件が必要です。

- KMS キーの [キーストア](#) は Enabled である必要があります。キーステータスを検索するには、[AWS KMS コンソール](#) のカスターマネージドキーの [Status] (ステータス) フィールド、または [DescribeKey](#) レスポンスの KeyState フィールドを参照してください。
- KMS キーをホストする外部キーストアは、その [外部キーストアプロキシ](#) に接続する必要があります。つまり、外部キーストアの [接続状態](#) は CONNECTED である必要があります。

接続状態は、AWS KMS コンソールの外部キーストアページまたは [DescribeCustomKeyStores](#) レスポンスで確認できます。外部キーストアの接続状態は、AWS KMS コンソールの KMS キーの詳細ページにも表示されます。詳細ページで [Cryptographic configuration] (暗号化設定) タブを選択し、[Custom key store] (カスタムキーストア) セクションの [Connection state] (接続状態) フィールドを確認します。

接続状態が DISCONNECTED の場合、最初に接続する必要があります。接続状態が FAILED の場合、問題を解決してから外部キーストアを切断し、接続する必要があります。手順については、「[外部キーストアを接続および切断する](#)」を参照してください。

- 外部キーストアプロキシが外部キーを検出する必要があります。
- 外部キーを有効にして、暗号化と復号を実行する必要があります。

外部キーのステータスは、KMS キーの有効化や無効化など、KMS キーの [キーステータス](#) の変化とは無関係で、影響を受けません。同様に、外部キーを無効化または削除しても KMS キーのキーステータスは変わりませんが、関連する KMS キーを使用する暗号化オペレーションは失敗します。

これらの条件が満たされない場合、暗号化オペレーションは失敗し、KMSInvalidStateException 例外 AWS KMS を返します。[外部キーストアを再接続](#)するか、外部キーマネージャツールを使用して、外部キーを再設定または修復する必要がある場合があります。その他のヘルプについては、「[the section called “外部キーストアのトラブルシューティング”](#)」を参照してください。

外部キーストアで KMS キーを使用する場合、各外部キーストアの KMS キーは、暗号化オペレーションで [カスタムキーストアのリクエストクォータ](#) を共有することに注意してください。クォー

タを超えると、は AWS KMS を返します `ThrottlingException`。カスタムキーストアのリクエストクォータの詳細については、「[カスタムキーストアのリクエストクォータ](#)」を参照してください。

詳細はこちら

- 外部キーストアの詳細については、「[外部キーストア](#)」を参照してください。
- 外部キーストアのキーマテリアルの詳細については、「[外部キー](#)」を参照してください。
- 外部キーストアで KMS キーを作成するには、「[外部キーストアで KMS キーを作成する](#)」を参照してください。
- 外部キーストアで KMS キーを特定して表示するには、「[外部キーストアの KMS キーを特定する](#)」を参照してください。
- 外部キーストアで KMS キーを削除する際の注意事項については、「[外部キーストアからの KMS キーの削除](#)」を参照してください。

AWS KMS 暗号化の必須事項

AWS KMS は設定可能な暗号化アルゴリズムを使用するため、システムは承認されたアルゴリズムまたはモードから別のアルゴリズムにすばやく移行できます。初期の暗号化アルゴリズムのデフォルトセットは、安全性とパフォーマンスのため、連邦情報処理標準 (FIPS 承認) のアルゴリズムから選択されています。

推奨および許容可能な暗号化アルゴリズムの詳細については、「」を参照してください [サポートされている暗号アルゴリズム](#)。

エントロピーと乱数生成

AWS KMS キーの生成は AWS KMS HSMs で実行されます。HSM では、[AES-256 を使った NIST SP800-90A 決定論的ランダムビットジェネレーター \(DRBG\) CTR_DRBG](#) を使用したハイブリッド乱数生成機能を実装しています。非決定論的ランダムビットジェネレーターは 384 ビットのエントロピーでシードされ、追加のエントロピーで更新されます。これにより、暗号化マテリアルの呼び出しごとに予測抵抗が提供されます。

対称キーのオペレーション (暗号化のみ)

HSM 内で使用されるすべての対称キーの暗号化コマンドには、[高度暗号化規格 \(AES\)](#) で 256 ビットキーの [Galois Counter Mode \(GCM\)](#) を使用します。復号化のための同様の呼び出しには、逆関数を使用します。

AES-GCM は認証済みの暗号化スキームです。プレーンテキストの暗号化による暗号文の生成に加えて、暗号文と認証が必要な追加データ (追加で認証されたデータ、または AAD) に対する認証タグを計算します。認証タグは、データが意図されたソースからのものであり、暗号文および AAD が変更されていないことを確認するのに役立ちます。

多くの場合、特にデータキーの暗号化を参照する場合、は説明に AAD を含めることを AWS 省略します。これらのケースでは周囲のテキストにより、暗号化される構造が、暗号化されるプレーンテキストと保護されるクリアテキスト AAD の間で分割されることが暗示されます。

AWS KMS には、を使用してキーマテリアルを生成する AWS KMS key 代わりに、キーマテリアルを AWS KMS にインポートするオプションが用意されています。このインポートされたキーマテリアルは、[RSAES-OAEP](#) を使用して暗号化し、AWS KMS HSM への転送中にキーを保護できます。RSA キーペアは AWS KMS HSMs。インポートされたキーマテリアルは AWS KMS HSM で復号され、サービスによって保存される前に AES-GCM で再暗号化されます。

非対称キーのオペレーション (暗号化、デジタル署名、署名の検証)

AWS KMS は、暗号化、デジタル署名、キーアグリーメントオペレーションの両方で非対称キーオペレーションの使用をサポートしています。非対称キーオペレーションは、数学的に関連付けられるパブリックキーとプライベートキーのペアに依存します。これらは暗号化と復号化、署名と検証、および共有シークレットの取得のいずれかの用途に使用できますが、一つのキーペアを複数の用途に使用することはできません。プライベートキーが暗号化 AWS KMS されていないままになることはありません。API AWS KMS オペレーションを呼び出し AWS KMS で内でパブリックキーを使用するか、パブリックキーをダウンロードしての外部で使用できます AWS KMS。

AWS KMS は、次の非対称暗号をサポートしています。

- RSA-OAEP (暗号化用) および RSA-PSS/RSA-PKCS-#1-v1_5 (署名および検証用) – さまざまなセキュリティ要件に対応できるように、RSA キーの長さは 2048、3072、4096 (ビット単位) がサポートされています。
- 楕円曲線 (ECC) – 署名と検証、もしくは共有シークレットの取得のいずれかの用途に使用できますが、両方の用途に使用することはできません。ECC 曲線 NIST P256、P384、P521、SECP 256k1 がサポートされています。
- ML-DSA – 署名と検証に使用されます。サポートされている ML-DSA キー仕様は、ML_DSA_44、ML_DSA_65、および ML_DSA_87 です。
- SM2 (中国リージョンのみ) – 暗号化と復号、署名と検証、もしくは共有シークレットの取得のいずれかの用途に使用できますが、いずれか一つの用途を選択する必要があります。暗号化用の SM2PKE と署名用の SM2DSA をサポートします。

キー導出関数

キー取得関数は、最初のシークレットまたはキーから追加のキーを取得するために使用されます。では、キー取得関数 (KDF) AWS KMS を使用して、データの暗号化ごとにコールごとのキーを取得します AWS KMS key。すべての KDF オペレーションでは、SHA256 [\[FIPS180\]](#) で HMAC [\[FIPS197\]](#)、[カウンタモードの KDF](#) を使用します。256 ビットの導出キーは AES-GCM とともに使用され、顧客データとキーを暗号化または復号化します。

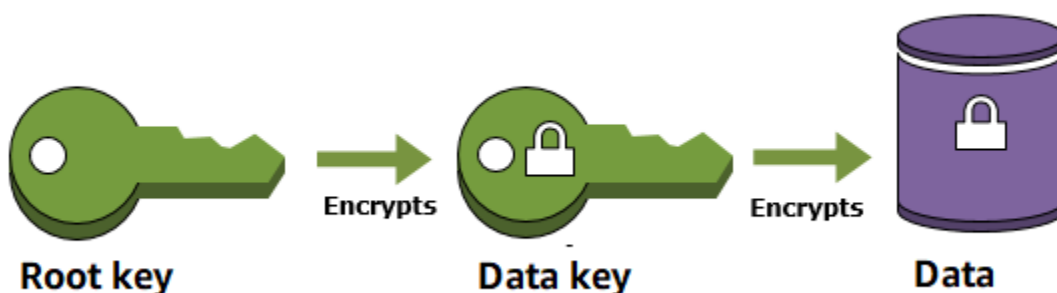
AWS KMS デジタル署名の内部使用

デジタル署名は、AWS KMS エンティティ間のコマンドや通信を認証するためにも使用されます。すべてのサービスエンティティには、楕円曲線デジタル署名アルゴリズム (ECDSA) のキーペアがあります。[Use of Elliptic Curve Cryptography \(ECC\) Algorithms in Cryptographic Message Syntax \(CMS\)](#) および X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA) で定義されているように、これらにより ECDSA が実行されます。エンティティでは、[Federal Information Processing Standards Publications, FIPS PUB 180-4](#) で定義されている、SHA384 と呼ばれる安全なハッシュアルゴリズムを使用します。キーは、曲線 secp384r1 (NIST-P384) 上で生成されます。

エンベロープ暗号化

データを暗号化するとデータは保護されますが、暗号化キーを保護する必要があります。1 つの方法としては、それを暗号化します。エンベロープ暗号化は、データキーでプレーンテキストデータを暗号化してから、そのデータキーを別のキーで暗号化する手法です。

データ暗号化キーを別の暗号化キーで暗号化し、その暗号化キーを別の暗号化キーで暗号化することもできます。しかし、最終的には、キーとデータを復号するために、1 つのキーをプレーンテキストで保持する必要があります。この最上位プレーンテキストキーの暗号化キーは、ルートキーと呼ばれます。



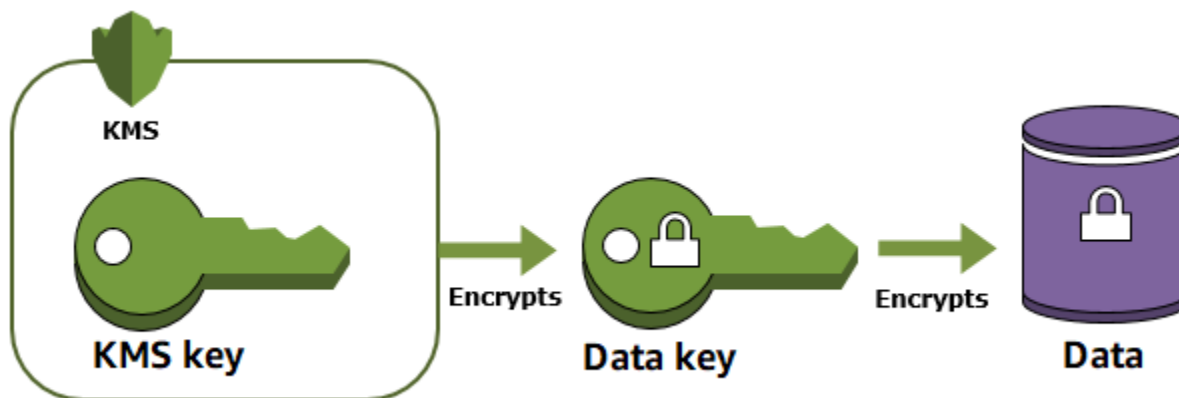
AWS KMS は、暗号化キーを安全に保存および管理することで、暗号化キーを保護するのに役立ちます。に保存されているルートキーは AWS KMS、AWS KMS [FIPS 140-3 セキュリティレベル 3 で検証されたハードウェアセキュリティモジュール](#)を暗号化したままに AWS KMS keys しないでください。KMS キーを使用するには、 を呼び出す必要があります AWS KMS。

エンベロープ暗号化は、多くの暗号化システムで使用されている基本的な構造です。エンベロープ暗号化では、2 つ以上の暗号化キーを使用してメッセージを保護します。通常、キーのうち 1 つはより長期的な静的キー k から取得します。もう 1 つはメッセージの暗号化のために生成されるメッセージごとのキー $msgKey$ です。エンベロープは、メッセージ $ciphertext = \text{Encrypt}(msgKey, message)$ の暗号化により形成されます。その後、長期的な静的キー $encKey = \text{Encrypt}(k, msgKey)$ でメッセージキーが暗号化されます。最後に、2 つの値 $(encKey, ciphertext)$ が、1 つの構造またはエンベロープで暗号化されたメッセージにパッケージ化されます。

k へのアクセス権を持つ受信者は、最初に暗号化されたキーを復号化してからメッセージを復号化することで、エンベロープされたメッセージを開くことができます。

AWS KMS は、これらの長期的な静的キーを管理し、データのエンベロープ暗号化のプロセスを自動化する機能を提供します。

[AWS Encryption SDK](#) は、AWS KMS サービス内で提供される暗号化機能に加えて、クライアント側のエンベロープ暗号化ライブラリを提供します。これらのライブラリを使用して、データとその暗号化に使用している暗号化キーを保護できます。



エンベロープ暗号化には、いくつかの利点があります。

- データキーの保護

データキーを暗号化する場合、暗号化されたデータキーの保存について心配する必要がありません。これは、そのデータキーが暗号化によって本質的に保護されているためです。暗号化されたデータとともに、暗号化されたデータキーを安全に保存できます。

- 複数のキーを使用した同じデータの暗号化

暗号化オペレーションには時間がかかります (特に、暗号化するデータが大きいオブジェクトである場合)。異なるキーで raw データを複数回にわたって再暗号化する代わりに、raw データを保護するデータキーのみを再暗号化できます。

- 複数のアルゴリズムの強度の結合

一般に、対称キーアルゴリズムは、パブリックキーアルゴリズムよりも高速で、より小さい暗号文を生成します。一方、パブリックキーのアルゴリズムはロールを本質的に分離し、キー管理を簡単にします。エンベロープ暗号化により、それぞれの方法を組み合わせることができます。

暗号化オペレーション

では AWS KMS、暗号化オペレーションは KMS キーを使用してデータを保護する API オペレーションです。KMS キーは 内に留まるため AWS KMS、暗号化オペレーションで KMS キー AWS KMS を使用するには を呼び出す必要があります。

KMS キーを使用して暗号化オペレーションを実行するには、AWS SDKs、(AWS CLI)、AWS Command Line Interface または を使用します AWS Tools for PowerShell。AWS KMS コンソールで暗号化オペレーションを実行することはできません。いくつかのプログラミング言語で暗号化オペレーションを呼び出す例については、を参照してください [AWS SDKs AWS KMS を使用するためのコード例](#)。

次の表に AWS KMS、暗号化オペレーションを示します。また、オペレーションで使用される KMS キーのキータイプと [キー使用法](#) の要件も示します。

オペレーション	キーのタイプ	キーの用途
Decrypt	対称または非対称	ENCRYPT_DECRYPT
DeriveSharedSecret	非対称	KEY_AGREEMENT
暗号化	対称または非対称	ENCRYPT_DECRYPT
GenerateDataKey	対称	ENCRYPT_DECRYPT
GenerateDataKeyPair	非対称 [1]	ENCRYPT_DECRYPT

オペレーション	キーのタイプ	キーの用途
	カスタムキーストアの KMS キーではサポートされません。	
GenerateDataKeyPairWithoutPlaintext	非対称 [1] カスタムキーストアの KMS キーではサポートされません。	ENCRYPT_DECRYPT
GenerateDataKeyWithoutPlaintext	対称	ENCRYPT_DECRYPT
GenerateMac	HMAC	GENERATE_VERIFY_MAC
GenerateRandom	該当なし。このオペレーションでは KMS キーを使用しません。	該当なし
ReEncrypt	対称または非対称	ENCRYPT_DECRYPT
Sign	非対称	SIGN_VERIFY
検証	非対称	SIGN_VERIFY
VerifyMac	HMAC	GENERATE_VERIFY_MAC

[1] 対称暗号化 KMS キーによって保護される非対称データキーペアを生成します。

暗号化オペレーションのアクセス許可については、「[the section called “アクセス許可に関するリファレンス”](#)」を参照してください。

すべてのユーザーに対して AWS KMS 応答性と高機能を実現するために、は 1 秒あたりに呼び出される暗号化オペレーションの数にクォータ AWS KMS を確立します。詳細については、[the section called “暗号化オペレーションの共有クォータ”](#) を参照してください

サポートされている暗号アルゴリズム

次の表 AWS は、データを保護するためにサービス全体にデプロイされている暗号化アルゴリズム、暗号、モード、およびキーサイズをまとめたものです。これは、で使用されるすべての暗号化の完全なリストとはみなされません AWS。アルゴリズムは 2 つのカテゴリに分類されます。「優先」は業界標準を満たし、[相互運用性](#)を促進するアルゴリズムであり、「許容可能」は特定のアプリケーションの互換性に使用できますが、推奨されません。暗号化ユースケースで暗号化を選択するときは、次の情報を考慮する必要があります。

にデプロイされる暗号化アルゴリズムの詳細については AWS、[「Cryptography algorithms and AWS のサービス」](#)を参照してください。

非対称暗号化

アルゴリズム	ステータス
2048 または 3072 ビットモジュラスを使用する RSA-OAEP	Acceptable
P-256 または P-384、HKDF、AES-GCM を使用した HPKE	Acceptable

非対称キーアグリーメント

アルゴリズム	ステータス
ML-KEM-768 または ML-KEM-1024	優先 (量子耐性用)
P-384 を使用した ECDH(E)	Acceptable
P-256, P-521 または X25519 を使用した ECDH(E)	Acceptable
ECDH(E) と brainpoolP256r1, brainpoolP384r1、または brainpoolP512r1	Acceptable

[署名]

アルゴリズム	ステータス
ML-DSA-65 または ML-DSA-87	優先 (量子耐性用)
SLH-DSA	推奨 (量子耐性ソフトウェア/ファームウェア署名用)
P-384 を使用した ECDSA	Acceptable
P-256, P-521、または Ed25519 を使用した ECDSA	Acceptable
RSA-2048 または RSA-3072	Acceptable

認証されている暗号化

アルゴリズム	ステータス
AES-GCM-256	推奨値
AES-GCM-128	Acceptable
ChaCha20/Poly1305	Acceptable

暗号化モード

アルゴリズム	ステータス
AES-XTS-256 (ブロックストレージ用)	推奨値
AES-CBC/CTR (認証されていないモード)	Acceptable

ハッシュ

アルゴリズム	ステータス
SHA2-384	推奨値

アルゴリズム	ステータス
SHA2-256	Acceptable
SHA3	Acceptable

キー取得

アルゴリズム	ステータス
SHA2-256 を使用した HKDF_Expand または HKDF	推奨値
HMAC-SHA2-256 を使用したカウンターモード KDF	Acceptable

キーラップ

アルゴリズム	ステータス
AES-GCM-256	推奨値
AES-KW または AES-KWP と 256 ビットキー	Acceptable

メッセージ認証コード (MAC)

アルゴリズム	ステータス
HMAC-SHA2-384	推奨値
HMAC-SHA2-256	Acceptable
KMAC	Acceptable

パスワードハッシュ

アルゴリズム	ステータス
SHA384 による暗号化	推奨値
PBKDF2	Acceptable

KMS キーのアクセスとアクセス許可

を使用するには AWS KMS、 がリクエストの認証 AWS に使用できる認証情報が必要です。認証情報には、AWS リソース AWS KMS keys と [エイリアス](#) にアクセスするためのアクセス許可が含まれている必要があります。アクセス許可が明示的に提供され、拒否されていない限り、KMS キーに対するアクセス許可を持つ AWS プリンシパルはありません。KMS キーを使用または管理するための暗黙的または自動的な権限はありません。

KMS キーへのアクセスを制御するには、次のポリシーメカニズムを使用できます。

- [キーポリシー](#) — すべての KMS キーにはキーポリシーがあります。これは、KMS キーへのアクセスを制御するための主要メカニズムです。キーポリシーを単独で使用してアクセスを制御できます。この場合、KMS キーへのアクセスの完全な範囲が 1 つのドキュメント (キーポリシー) で定義されます。キーポリシーの使用の詳細については、「[キーポリシー](#)」を参照してください。
- [IAM ポリシー](#) — IAM ポリシーをキーポリシーと組み合わせて使用し、KMS キーへのアクセスを制御できるように許可を付与します。この方法でアクセスを制御すると、IAM の IAM ID に対するすべてのアクセス許可を管理できます。IAM ポリシーを使用して KMS キーへのアクセスを許可するには、キーポリシーで明示的に許可する必要があります。IAM ポリシーの使用の詳細については、「[IAM ポリシー](#)」を参照してください。
- [グラント](#) — グラントをキーポリシーおよび IAM ポリシーと組み合わせて使用し、KMS キーへのアクセスを許可します。この方法でアクセスを制御すると、キーポリシーで KMS キーへのアクセスを許可し、アイデンティティが他のユーザーにアクセスを委任できるようになります。グラントの使用の詳細については、「[の許可 AWS KMS](#)」を参照してください。

KMS キーポリシー

AWS KMS リソースへのアクセスを管理する主な方法は、ポリシーを使用することです。ポリシーは、どのプリンシパルがどのリソースにアクセスできるかを記述するドキュメントです。IAM アイデンティティにアタッチされたポリシーはアイデンティティベースのポリシー (IAM ポリシー) と呼ばれ、他の種類のリソースにアタッチされたポリシーはリソースポリシーと呼ばれます。KMS キーの AWS KMS リソースポリシーはキーポリシーと呼ばれます。

すべての KMS キーにはキーポリシーがあります。指定しない場合、によって AWS KMS 自動的に作成されます。が AWS KMS 使用する [デフォルトのキーポリシー](#) は、AWS KMS コンソールでキーを作成するか AWS KMS API を使用するかによって異なります。デフォルトのキーポリシーを編集して、[最小特権のアクセス許可](#) に関する組織の要件に合わせることをお勧めします。

キーと IAM プリンシパルが同じ AWS アカウントにある場合、キーポリシーのみを使用してアクセスを制御できます。つまり、KMS キーへのアクセスの全範囲は 1 つのドキュメント (キーポリシー) で定義されます。ただし、あるアカウントの発信者が別のアカウントのキーにアクセスする場合には、キーポリシーのみでアクセスを許可することはできません。複数アカウントにまたがるシナリオにおいては、呼び出し元が API コールを行うことを明示的に許可する IAM ポリシーを呼び出し元のユーザーまたはロールにアタッチする必要があります。

また、IAM ポリシーをキーポリシーと組み合わせて使用して、KMS キーへのアクセス制御を許可することもできます。IAM ポリシーを使用して KMS キーへのアクセスを制御するには、IAM ポリシーを使用するアクセス許可を、キーポリシーによりアカウントに付与する必要があります。[IAM ポリシーを有効にするキーポリシーステートメント](#)を指定するか、もしくはキーポリシー内で[許可対象のプリンシパルを明示的に指定](#)できます。

ポリシーを作成するユーザーは、次のアクションをどのユーザーが実行できるかを制限できる十分なコントロールを備えている必要があります。

- IAM および KMS キーポリシーの更新、作成、削除
- ユーザー、ロール、およびグループからの IAM ポリシーのアタッチおよびデタッチ
- KMS キーからの KMS キーポリシーのアタッチとデタッチ

KMS キー向けグラント

IAM ポリシーとキーポリシーに加えて、は[許可](#) AWS KMS をサポートします。グラントは、アクセス許可付与のためのフレキシブルかつ強力な手法です。権限を使用して、アカウントまたは他の AWS アカウントの IAM プリンシパルに期限付き KMS キーアクセスを発行できます AWS。ポリシーの作成時にプリンシパルの名前がわからない場合、あるいはアクセスを必要とするプリンシパルが頻繁に変更される場合は、時間制限付きアクセスを発行することをお勧めします。[グラント対象のプリンシパル](#)は、KMS キーと同じアカウントにあっても、あるいは別のアカウントにあっても良いです。プリンシパルキーと KMS キーが異なるアカウントにある場合は、グラントに加えて IAM ポリシーを指定する必要があります。グラントを作成するには API を呼び出し、不要になったグラントを廃止または取り消す必要があるため、グラントには追加の管理が必要です。

のキーポリシー AWS KMS

キーポリシーは、のリソースポリシーです AWS KMS key。キーポリシーは、KMS キーへのアクセスを制御するための主要な方法です。すべての KMS キーには、厳密に 1 つのキーポリシーが必要です。キーポリシーのステートメントでは、KMS キーの使用が許可されるユーザーとその使用方法を

決定します。[IAM ポリシー](#)と[グラント](#)を使用して KMS キーへのアクセスを制御することもできますが、すべての KMS キーにはキーポリシーが必要です。

アカウントのルートユーザーまたはキー作成者を含むプリン AWS シパルには、キーポリシー、IAM ポリシー、または許可で明示的に許可され、拒否されていない限り、KMS キーに対するアクセス許可はありません。

キーポリシーで明示的に許可されていない限り、IAM ポリシーを使用して KMS キーへのアクセスを許可することはできません。キーポリシーからの許可がない場合、許可を許可する IAM ポリシーは効力を持ちません。(IAM ポリシーを使用して、キーポリシーからの許可を得ることなく KMS キーに対する許可を拒否することができます。) デフォルトのキーポリシーでは IAM ポリシーが有効になっています。キーポリシーで IAM ポリシーを有効にするには、[AWS アカウント へのアクセスを許可し、IAM ポリシーを有効にする](#) で説明されているポリシーステートメントを追加します。

グローバルな IAM ポリシーとは異なり、キーポリシーはリージョンナルです。キーポリシーは、同じリージョン内の KMS キーへのアクセスのみを制御します。他のリージョンの KMS キーには影響しません。

トピック

- [キーポリシーを作成する](#)
- [デフォルトのキーポリシー](#)
- [キーポリシーを表示する](#)
- [キーポリシーを変更する](#)
- [キーポリシーでの AWS サービスのアクセス許可](#)

キーポリシーを作成する

キーポリシーは、AWS KMS コンソールで、または [CreateKey](#)、[ReplicateKey](#)、[PutKeyPolicy](#) などの AWS KMS API オペレーションを使用して作成および管理できます。

AWS KMS コンソールで KMS キーを作成すると、コンソールはコンソールの[デフォルトキーポリシーに基づいてキーポリシー](#)を作成する手順を説明します。CreateKey または ReplicateKey API を使用するときにはキーポリシーを指定しない場合、これらの API は[プログラマ的に作成されたキーのデフォルトキーポリシー](#)を適用します。PutKeyPolicy API を使用するときには、キーポリシーを指定する必要があります。

各ポリシードキュメントには、1 つ以上のポリシーステートメントを指定できます。以下は、1 つのポリシーステートメントがある有効なキーポリシードキュメントの例です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DescribePolicyStatement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/Alice"
      },
      "Action": "kms:DescribeKey",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:KeySpec": "SYMMETRIC_DEFAULT"
        }
      }
    }
  ]
}
```

トピック

- [キーポリシー形式](#)
- [キーポリシー内の要素](#)
- [キーポリシーの例](#)

キーポリシー形式

キーポリシードキュメントは、以下のルールに従う必要があります。

- 最大 32 キロバイト (32,768 バイト)
- キーポリシーステートメントの Sid 要素には、スペースを含めることができます。(IAM ポリシードキュメントの Sid 要素では、スペースが禁止されています。)

キーポリシードキュメントには、以下の文字のみを含めることができます。

- 印字可能な ASCII 文字

- 基本ラテンおよびラテン 1 補助文字セットの印字可能な文字
- タブ (\u0009)、ラインフィード (\u000A)、およびキャリッジリターン (\u000D) の特殊文字

キーポリシー内の要素

キーポリシードキュメントには、次の要素が必要です。

Version

キーポリシードキュメントのバージョンを指定します。バージョンを 2012-10-17 (最新バージョン) に設定します。

Statement

ポリシーステートメントを囲みます。キーポリシードキュメントには、少なくとも 1 つのステートメントが必要です。

各キーポリシーステートメントは、最大 6 個の要素で構成されます。Effect、Principal、Action、Resource の要素は必須です。

Sid

(オプション) ステートメント識別子 (Sid)。ステートメントを記述するために使用できる任意の文字列です。キーポリシーの Sid には、スペースを含めることができます。(IAM ポリシーの Sid 要素にスペースを含めることはできません。)

Effect

(必須) ポリシーステートメント内の許可を許容するか拒否するかを決定します。有効な値は Allow または Deny です。KMS キーへのアクセスを明示的に許可しない場合、アクセスは暗黙的に拒否されます。KMS キーへのアクセスを明示的に拒否することもできます。これは、別のポリシーがアクセスを許可している場合でも、ユーザーがアクセスできないようにするために行います。

Principal

(必須) [プリンシパル](#)は、ポリシーステートメントで指定されている許可を取得するアイデンティティです。キーポリシーでは AWS アカウント、プリンシパルとして、IAM ユーザー、IAM ロール、および一部の AWS サービスを指定できます。IAM [ユーザーグループ](#)は、どのポリシータイプにおいても有効なプリンシパルではありません。

"AWS": "*" などのアスタリスク値は、すべてのアカウントのすべての AWS ID を表します。

⚠ Important

[条件](#)を使用してキーポリシーを制限しない限り、アクセス許可を付与するキーポリシーステートメントで、プリンシパルをアスタリスク (*) に設定しないでください。アスタリスクは、別のポリシーステートメントが明示的に拒否しない限り、KMS キーを使用するすべての AWS アカウント アクセス許可をすべての ID に付与します。他のユーザーは AWS アカウント、自分のアカウントで対応するアクセス許可があるたびに KMS キーを使用できます。

i Note

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

キーポリシーステートメントのプリンシパルが `arn:aws:iam::111122223333:root` 形式の [AWS アカウント プリンシパル](#)である場合、ポリシーステートメントはいかなる IAM プリンシパルにも許可を付与しません。代わりに、IAM ポリシーを使用してキーポリシーで指定された AWS アカウント アクセス許可を委任するアクセス許可が 1 つに付与されます。(arn:aws:iam::**111122223333**:root 形式のプリンシパルは、アカウント識別子に「root」が使用されていても [AWS アカウントのルートユーザー](#)を示すものではありません。ただし、アカウントプリンシパルは、アカウントのルートユーザーを含むアカウントとその管理者を表します。)

プリンシパルが別のプリンシパル AWS アカウント またはそのプリンシパルである場合、アクセス許可は、KMS キーとキーポリシーを持つリージョンでアカウントが有効になっている場合にのみ有効です。デフォルトで有効になっていないリージョン (「[オプトインリージョン](#)」) については、「AWS 全般のリファレンス」の「[AWS リージョンの管理](#)」を参照してください。

別の AWS アカウント またはそのプリンシパルに KMS キーの使用を許可するには、キーポリシーと他のアカウントの IAM ポリシーでアクセス許可を付与する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

Action

(必須) 許可または拒否する API オペレーションを指定します。たとえば、`kms:Encrypt` アクションは AWS KMS [Encrypt](#) オペレーションに対応します。ポリシーステートメントに複数のアクションを一覧表示できます。詳細については、「[アクセス許可に関するリファレンス](#)」を参照してください。

Note

必要な Action 要素がキーポリシーステートメントにない場合、ポリシーステートメントによる影響はありません。Action 要素のないキーポリシーステートメントは、どの KMS キーにも適用されません。

キーポリシーステートメントに Action 要素がない場合、AWS KMS コンソールはエラーを正しく報告しますが、ポリシーステートメントが無効であっても [CreateKey](#) API と [PutKeyPolicy](#) APIs は成功します。

リソース

(必須) キーポリシーでは、Resource 要素の値が "*" になります。これは「この KMS キー」を意味します。アスタリスク ("*") は、キーポリシーがアタッチされた KMS キーを識別します。

Note

必要な Resource 要素がキーポリシーステートメントにない場合、ポリシーステートメントによる影響はありません。Resource 要素のないキーポリシーステートメントは、どの KMS キーにも適用されません。

キーポリシーステートメントに Resource 要素がない場合、AWS KMS コンソールはエラーを正しく報告しますが、ポリシーステートメントが無効であっても [CreateKey](#) API と [PutKeyPolicy](#) APIs は成功します。

Condition

(オプション) 条件は、キーポリシーを有効にするために満たす必要がある要件を指定します。条件を使用すると、は API リクエストのコンテキストを評価して、ポリシーステートメントが適用されるかどうかを判断 AWS できます。

条件を指定するには、事前定義された条件キーを使用します。は[AWS グローバル条件キー](#)と[AWS KMS 条件キー](#) AWS KMS をサポートしています。属性ベースのアクセスコントロール (ABAC) をサポートするために、はタグとエイリアスに基づいて KMS キーへのアクセスを制御する条件キー AWS KMS を提供します。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

条件の形式は次のとおりです。

```
"Condition": {"condition operator": {"condition key": "condition value"}}
```

例 :

```
"Condition": {"StringEquals": {"kms:CallerAccount": "111122223333"}}
```

AWS ポリシー構文の詳細については、IAM [AWS ユーザーガイドの「IAM ポリシーリファレンス」](#)を参照してください。

キーポリシーの例

以下は、対称暗号化 KMS キーの完全なキーポリシーの例です。この章の主なポリシー概念について読むと、参考に使うことができます。このキーポリシーでは、先ほどの「[デフォルトキーポリシー](#)」のセクションからのポリシーステートメント例と、以下を実行するための 1 つのキーポリシーを組み合わせています。

- 例 AWS アカウント 111122223333 に KMS キーへのフルアクセスを許可します。これにより、アカウントのルートユーザー (緊急用) を含む、アカウントとその管理者は、アカウントの IAM ポリシーを使用して KMS キーへのアクセスを許可できます。
- ExampleAdminRole IAM ロールが KMS キーを管理できるようにします。
- ExampleUserRole IAM ロールが KMS キーを使用できるようにします。

JSON

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

```

        "Sid": "EnableIAMUserPermissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
        },
        "Action": "kms:*",
        "Resource": "*"
    },
    {
        "Sid": "AllowKeyAdministratorsAccess",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/ExampleAdminRole"
        },
        "Action": [
            "kms:Create*",
            "kms:Describe*",
            "kms:Enable*",
            "kms:List*",
            "kms:Put*",
            "kms:Update*",
            "kms:Revoke*",
            "kms:Disable*",
            "kms:Get*",
            "kms>Delete*",
            "kms:TagResource",
            "kms:UntagResource",
            "kms:ScheduleKeyDeletion",
            "kms:CancelKeyDeletion",
            "kms:RotateKeyOnDemand"
        ],
        "Resource": "*"
    },
    {
        "Sid": "AllowKeyUse",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/ExampleUserRole"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",

```

```

        "kms:DescribeKey"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowAttachmentPersistentResources",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/ExampleUserRole"
    },
    "Action": [
      "kms:CreateGrant",
      "kms:ListGrants",
      "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
      "Bool": {
        "kms:GrantIsForAWSResource": "true"
      }
    }
  }
]
}

```

デフォルトのキーポリシー

KMS キーを作成するときに、新しい KMS キーのキーポリシーを指定できます。指定しない場合、によって AWS KMS 自動的に作成されます。が AWS KMS 使用するデフォルトのキーポリシーは、AWS KMS コンソールでキーを作成するか AWS KMS API を使用するかによって異なります。

KMS キーをプログラムにより作成したときのデフォルトキーポリシー

[AWS KMS API](#) を使用してプログラムで KMS キーを作成し ([AWS SDKs](#) [AWS Command Line Interface](#) または の使用を含む [AWS Tools for PowerShell](#))、キーポリシーを指定しない場合、は非常にシンプルなデフォルトキーポリシー AWS KMS を適用します。このデフォルトのキーポリシーには、IAM ポリシーを使用して KMS キー上のすべての AWS KMS オペレーションへのアクセスを許可するアクセス許可を KMS キーを所有 AWS アカウント する に付与する 1 つのポリシーステートメントがあります。このポリシーステートメントの詳細については、「[AWS アカウント へのアクセスを許可し、IAM ポリシーを有効にする](#)」を参照してください。

を使用して KMS キーを作成するときのデフォルトのキーポリシー AWS Management Console

[を使用して KMS キーを作成する AWS Management Console](#)と、キーポリシーは [へのアクセスを許可 AWS アカウント し、IAM ポリシーを有効にするポリシーステートメント](#)で始まります。次に、コンソールは、[キー管理者ステートメント](#)、[キーユーザーステートメント](#)、およびプリンシパルが[他の AWS サービス](#)で KMS キーを使用できるようにするステートメント (ほとんどのキータイプ) を追加します。AWS KMS コンソールの機能を使用して、IAM ユーザー、IAM ロール、キー管理者である AWS アカウント ユーザー、キーユーザー (またはその両方) を指定できます。

アクセス許可

- [AWS アカウント へのアクセスを許可し、IAM ポリシーを有効にする](#)
- [KMS キーの管理をキー管理者に許可する](#)
- [KMS キーの使用をキーユーザーに許可する](#)
 - [暗号化オペレーションで KMS キーを使用することをキーユーザーに許可する](#)
 - [AWS サービスで KMS キーを使用することをキーユーザーに許可する](#)

AWS アカウント へのアクセスを許可し、IAM ポリシーを有効にする

次のデフォルトのキーポリシーステートメントは重要です。

- これにより、KMS キーを所有 AWS アカウント する に KMS キーへのフルアクセスが付与されます。

他の AWS リソースポリシーとは異なり、AWS KMS キーポリシーはアカウントまたはそのアイデンティティに自動的にアクセス許可を付与しません。アカウント管理者に許可を付与するには、キーポリシーに、このような許可を提供する明示的なステートメントを含める必要があります。

- これにより、アカウントは IAM ポリシーを使用して、キーポリシーに加えて KMS キーへのアクセスを許可できるようになります。

この許可がないと、キーへのアクセスを許可する IAM ポリシーは無効になりますが、キーへのアクセスを拒否する IAM ポリシーは依然として有効です。

- 削除できないアカウントのルートユーザーを含むアカウント管理者にアクセスコントロールの許可を付与することで、キーが管理不能になるリスクが軽減されます。

次のキーポリシーステートメントは、プログラムで作成された KMS キーのデフォルトキーポリシーの全文です。これは、AWS KMS コンソールで作成された KMS キーのデフォルトキーポリシーの最初のポリシーステートメントです。

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:root"
  },
  "Action": "kms:*",
  "Resource": "*"
}
```

IAM ポリシーが KMS キーへのアクセスを許可できるようにします。

上記のキーポリシーステートメントは、IAM ポリシーとキーポリシーを使用して、KMS キーに対するすべてのアクション (kms:*) を許可するキーアクセス許可を所有 AWS アカウント するに付与します。

このキーポリシーステートメントのプリンシパルは [アカウントプリンシパル](#) であり、これはこの形式にある ARN で表されています: arn:aws:iam::**account-id**:root。アカウントプリンシパルは、AWS アカウントとその管理者を表します。

キーポリシーステートメントのプリンシパルがアカウントプリンシパルである場合、ポリシーステートメントはどの IAM プリンシパルに対しても、KMS キーを使用する許可を付与しません。代わりにアカウントに IAM ポリシーを使用して、ポリシーステートメントで指定された許可を委譲することができます。このデフォルトのキーポリシーステートメントにより、アカウントは IAM ポリシーを使用して、KMS キーに対するすべてのアクション (kms:*) の許可を委譲することができます。

KMS キーが管理不能になるリスクを減らします。

他の AWS リソースポリシーとは異なり、AWS KMS キーポリシーはアカウントまたはそのプリンシパルに自動的にアクセス許可を付与しません。プリンシパルに許可を与えるには ([アカウントプリンシパル](#)を含む)、許可を明示的に提供するキーポリシーステートメントを使用する必要があります。アカウントプリンシパルまたはプリンシパルに、KMS キーへのアクセス権を付与する必要はありません。ただし、アカウントプリンシパルにアクセス権を付与することは、キーが管理不能になるのを防ぐために役立ちます。

たとえば、KMS キーへのアクセスを 1 人のユーザーだけに付与するキーポリシーを作成したとします。その後、そのユーザーを削除すると、キーは管理不能になり、KMS キーへのアクセスを取り戻すために[AWS サポート](#)に連絡しなければなりません。

上記のキーポリシーステートメントは、アカウントの[ルートユーザー](#)を含む AWS アカウントとその管理者を表すアカウント[プリンシパル](#)にキーを制御するアクセス許可を付与します。アカウントのルートユーザーは、AWS アカウントを削除しない限り削除できない唯一のプリンシパルです。IAM のベストプラクティスでは、緊急時を除き、アカウントのルートユーザーの代わりとして行動することを推奨していません。ただし、KMS キーへのアクセス権を持つ他のすべてのユーザーおよびロールを削除した場合には、アカウントのルートユーザーとして行動する必要がある場合があります。

KMS キーの管理をキー管理者に許可する

コンソールによって作成されるデフォルトのキーポリシーでは、アカウントで IAM ユーザーとロールを選択し、それらをキー管理者にすることができます。このステートメントは、キー管理者ステートメントと呼ばれます。キー管理者には、KMS キーを管理するためのアクセス許可はありますが、[暗号化オペレーション](#)で KMS キーを使用するためのアクセス許可はありません。デフォルトビューやポリシービューで KMS キーを作成するときに、IAM ユーザーとロールをキー管理者のリストに追加できます。

Warning

キー管理者にはキーポリシーを変更して許可を作成するアクセス許可があるため、このポリシーで指定されていないアクセス AWS KMS 許可を自分自身や他の管理者に付与できます。タグとエイリアスを管理するアクセス権限を持つプリンシパルも、KMS キーへのアクセスを制御することができます。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

Note

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

次の例は、AWS KMS コンソールのデフォルトビューでのキー管理者ステートメントを示しています。

The screenshot shows the AWS KMS console interface. At the top, there are two tabs: 'Key policy' (selected) and 'Tags'. Below the tabs, there's a 'Key policy' section with a 'Switch to policy view' button. The main section is titled 'Key administrators' and includes a description: 'Choose the IAM users and roles who can administer this key through the KMS API. You might need to add additional permissions for the users or roles to administer this key from this console. [Learn more](#)'. There are 'Add' and 'Remove' buttons, and a search bar. Below the search bar is a table with columns: Name, Path, and Type. The table contains one row: 'ExampleAdminRole' with a path of '/' and type of 'Role'. At the bottom, there's a 'Key deletion' section with a checkbox 'Allow key administrators to delete this key' which is checked.

	Name	Path	Type
☐	ExampleAdminRole	/	Role

AWS KMS コンソールのポリシービューでのキー管理者ステートメントの例を次に示します。このキー管理者ステートメントは、単一リージョンの対称暗号化 KMS キー用です。

Note

AWS KMS コンソールは、ステートメント識別子のキーポリシーにキー管理者を追加します "Allow access for Key Administrators"。このステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

```
{  
  "Sid": "Allow access for Key Administrators",
```

```
"Effect": "Allow",
"Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleAdminRole"},
"Action": [
  "kms:Create*",
  "kms:Describe*",
  "kms:Enable*",
  "kms:List*",
  "kms:Put*",
  "kms:Update*",
  "kms:Revoke*",
  "kms:Disable*",
  "kms:Get*",
  "kms>Delete*",
  "kms:TagResource",
  "kms:UntagResource",
  "kms:ScheduleKeyDeletion",
  "kms:CancelKeyDeletion",
  "kms:RotateKeyOnDemand"
],
"Resource": "*"
}
```

最も一般的な KMS キーである単一リージョンの対称暗号化 KMS キーのデフォルトキー管理者ステートメントは、以下の許可を付与します。各アクセス許可の詳細については、「[AWS KMS アクセス許可](#)」を参照してください。

AWS KMS コンソールを使用して KMS キーを作成すると、コンソールは指定したユーザーとロールをキー管理者ステートメントの Principal 要素に追加します。

これらのアクセス許可の多くには、指定した動詞で始まるアクセス許可すべてを許可するワイルドカード文字 (*) が含まれています。その結果、新しい API オペレーション AWS KMS を追加すると、キー管理者は自動的にそれらを使用できます。新しいオペレーションを含めるために、キーポリシーを更新する必要はありません。キー管理者を固定された一連の API オペレーションに限定する場合は、[キーポリシーを変更](#)することができます。

kms:Create*

[kms:CreateAlias](#) および [kms:CreateGrant](#) を許可します。(kms:CreateKey のアクセス許可は IAM ポリシーでのみ有効です)

kms:Describe*

[kms:DescribeKey](#) を許可します。– AWS Management Console で KMS キーのキーの詳細ページを表示するには、`kms:DescribeKey` のアクセス許可が必要です。

kms:Enable*

[kms:EnableKey](#) を許可します。対称暗号化 KMS キーについては、[kms:EnableKeyRotation](#) も許可します。

kms:List*

[kms:ListGrants](#)、[kms:ListKeyPolicies](#)、および [kms:ListResourceTags](#) を許可します。(KMS キーを AWS Management Console で表示するのに必要な `kms:ListAliases` および `kms:ListKeys` のアクセス許可は、IAM ポリシーでのみ有効です)

kms:Put*

[kms:PutKeyPolicy](#) を許可します。このアクセス許可により、キー管理者が、この KMS キーのキーポリシーを変更できるようになります。

kms:Update*

[kms:UpdateAlias](#) および [kms:UpdateKeyDescription](#) を許可します。マルチリージョンキーの場合、この KMS キーで [kms:UpdatePrimaryRegion](#) が許可されます。

kms:Revoke*

[kms:RevokeGrant](#) を許可します。これにより、キー管理者は自分がグラントの[無効なプリンシパル](#)でなくても[グラントを削除](#)できるようになります。

kms:Disable*

[kms:DisableKey](#) を許可します。対称暗号化 KMS キーについては、[kms:DisableKeyRotation](#) も許可します。

kms:Get*

[kms:GetKeyPolicy](#) および [kms:GetKeyRotationStatus](#) を許可します。インポートしたキーマテリアルを含む KMS キーの場合、[kms:GetParametersForImport](#) を許可します。非対称 KMS キーの場合、[kms:GetPublicKey](#) を許可します。AWS Management Console で KMS キーのキーポリシーを表示するには、`kms:GetKeyPolicy` のアクセス許可が必要です。

kms:Delete*

[kms:DeleteAlias](#) を許可します。インポートしたキーマテリアルを含むキーの場合、[kms:DeleteImportedKeyMaterial](#) を許可します。kms:Delete* のアクセス許可では、キー管理者が KMS キー (ScheduleKeyDeletion) を削除できません。

kms:TagResource

[kms:TagResource](#) を許可します。これにより、キー管理者は KMS キーにタグを追加できるようになります。タグは KMS キーへのアクセス制御にも使用できるため、このアクセス許可により、管理者が KMS キーへのアクセスを許可または拒否できるようになります。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

kms:UntagResource

[kms:UntagResource](#) を許可します。これにより、キー管理者は KMS キーからタグを削除できるようになります。タグはキーへのアクセス制御に使用できるため、このアクセス許可により、管理者は KMS キーへのアクセスを許可または拒否できるようになります。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

kms:ScheduleKeyDeletion

[kms:ScheduleKeyDeletion](#) を許可します。これにより、キー管理者は[この KMS キーを削除](#)できるようになります。このアクセス許可を削除するには、[Allow key administrators to delete this key] (キーの管理者がこのキーを削除できるようにします) オプションをオフにします。

kms:CancelKeyDeletion

[kms:CancelKeyDeletion](#) を許可します。これにより、キー管理者は[この KMS キーの削除をキャンセル](#)できるようになります。このアクセス許可を削除するには、[Allow key administrators to delete this key] (キーの管理者がこのキーを削除できるようにします) オプションをオフにします。

kms:RotateKeyOnDemand

[kms:RotateKeyOnDemand](#) を許可します。これによりキー管理者は、[この KMS キーのキーマテリアルのオンデマンドローテーションを実行できます](#)。

AWS KMS は、専用キーを作成するときに、デフォルトのキー管理者ステートメントに次のアクセス許可を追加します。

kms:ImportKeyMaterial

[kms:ImportKeyMaterial](#) のアクセス許可により、キー管理者が KMS キーにキーマテリアルをインポートできるようになります。このアクセス許可は、[キーマテリアルのない KMS キーを作成する](#)場合にのみキーポリシーに含まれます。

kms:ReplicateKey

アクセス[kms:ReplicateKey](#)許可により、キー管理者は別の[リージョンにマルチリージョンのプライマリキーのレプリカを作成できます](#) AWS。このアクセス許可は、マルチリージョンプライマリキーまたはマルチリージョンレプリカキーを作成する場合にのみ、キーポリシーに含まれます。

kms:UpdatePrimaryRegion

[kms:UpdatePrimaryRegion](#) のアクセス許可により、キー管理者が、[マルチリージョンレプリカキーをマルチリージョンプライマリキーに変更](#)できるようになります。このアクセス許可は、マルチリージョンプライマリキーまたはマルチリージョンレプリカキーを作成する場合にのみ、キーポリシーに含まれます。

KMS キーの使用をキーユーザーに許可する

コンソールが KMS キー用に作成するデフォルトのキーポリシーでは、アカウントおよび外部で IAM ユーザーと IAM ロールを選択し、キーユーザーに AWS アカウントすることができます。

コンソールは、キーユーザーのキーポリシーに 2 つのポリシーステートメントを追加します。

- [KMS キーの直接使用](#) — 最初のキーポリシーステートメントは、そのタイプの KMS キーでサポートされるすべての[暗号化オペレーション](#)に対して KMS キーを直接使用するアクセス許可をキーユーザーに付与します。
- [サービスで AWS KMS キー](#)を使用する — 2 番目のポリシーステートメントは、と統合されている AWS サービスがユーザーに代わって KMS キー AWS KMS を使用して Amazon S3 バケットや Amazon DynamoDB テーブルなどのリソースを保護するアクセス許可をキーユーザーに付与します。

KMS キーを作成するときに、IAM ユーザー、IAM ロールなどをキーユーザー AWS アカウント のリストに追加できます。次の図に示すように、キーポリシーのコンソールのデフォルトビューを使用してリストを編集することもできます。キーポリシー用デフォルトビューは、キーの詳細ページにあります。他の のユーザーに KMS キー AWS アカウント の使用を許可する方法の詳細については、「」を参照してください[他のアカウントのユーザーに KMS キーの使用を許可する](#)。

Note

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

Key users

The following IAM users and roles can use this key for cryptographic operations. They can also allow AWS services that are integrated with KMS to use the key on their behalf. [Learn more](#)

AddRemove

< 1 >

☐	Name	Path	Type
☐	ExampleRole	/	Role

Other AWS accounts

- arn:aws:iam::444455556666:root

デフォルトの単一リージョンの対称用キーユーザーステートメントにより、次のアクセス許可が付与されます。各アクセス許可の詳細については、[AWS KMS アクセス許可](#) を参照してください。

AWS KMS コンソールを使用して KMS キーを作成すると、コンソールは指定したユーザーとロールを各キーユーザーステートメントの Principal 要素に追加します。

Note

AWS KMS コンソールは、ステートメント識別子 "Allow use of the key" およびのキーポリシーにキーユーザーを追加します "Allow attachment of persistent

resources"。これらのステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
},
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

暗号化オペレーションで KMS キーを使用することをキーユーザーに許可する

キーユーザーには、KMS キーでサポートされるすべての[暗号化オペレーション](#)で KMS キーを直接使用するためのアクセス許可が付与されています。また、[DescribeKey](#) オペレーションを使用して、AWS KMS コンソールまたは AWS KMS API オペレーションを使用して KMS キーに関する詳細情報を取得することもできます。

デフォルトでは、AWS KMS コンソールは、次の例のようなキーユーザーステートメントをデフォルトのキーポリシーに追加します。これらは異なる API オペレーションをサポートするため、対称暗号化 KMS キー、HMAC KMS キー、パブリックキー暗号化用の非対称 KMS キー、および署名と検証用の非対称 KMS キー向けのポリシーステートメント内のアクションは、それぞれわずかに異なります。

対称暗号化 KMS キー

コンソールは、対称暗号化 KMS キーのキーポリシーに以下のステートメントを追加します。

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey",
    "kms:Encrypt",
    "kms:GenerateDataKey*",
    "kms:ReEncrypt*"
  ],
  "Resource": "*"
}
```

HMAC KMS キー

コンソールは、HMAC KMS キーのキーポリシーに以下のステートメントを追加します。

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GenerateMac",
    "kms:VerifyMac"
  ],
  "Resource": "*"
}
```

非対称 KMS キーのパブリックキー暗号化

コンソールでは、暗号化と復号のキーを使用して、非対称 KMS キーのキーポリシーに次のステートメントを追加します。

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:DescribeKey",
    "kms:GetPublicKey"
  ],
  "Resource": "*"
}
```

非対称 KMS キーの署名および検証

コンソールは、署名と検証のキーを使用して、非対称 KMS キーのキーポリシーに次のステートメントを追加します。

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:Sign",
    "kms:Verify"
  ],
  "Resource": "*"
}
```

共有シークレットを取得するための非対称 KMS キー

コンソールでは、[キーアグリーメント] のキーを使用して、非対称 KMS キーのキーポリシーに次のステートメントを追加します。

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:DeriveSharedSecret"
  ],
  "Resource": "*"
}
```

これらのステートメントのアクションは、キーユーザーに次のアクセス許可を付与します。

[kms:Encrypt](#)

キーユーザーがこの KMS キーでデータを暗号化できるようにします。

[kms:Decrypt](#)

キーユーザーがこの KMS キーでデータを復号できるようにします。

[kms:DeriveSharedSecret](#)

キーユーザーがこの KMS キーを使用して共有シークレットを取得できるようにします。

[kms:DescribeKey](#)

この KMS キーに関する詳細情報 (その識別子、作成日、キーステータスなど) を、キーユーザーが取得できるようにします。また、キーユーザーは KMS キーの詳細を AWS KMS コンソールに表示できます。

kms:GenerateDataKey*

クライアント側の暗号化オペレーションのために、対称データキーまたは非対称データキーペアをキーユーザーがリクエストできるようにします。コンソールは * ワイルドカード文字を使用して、次の API オペレーションの権限を表します:

[GenerateDataKey](#)、[GenerateDataKeyWithoutPlaintext](#)、[GenerateDataKeyPair](#)、[GenerateDataKeyPairWithoutPlaintext](#)

これらのアクセス許可は、データキーを暗号化する対称 KMS キーでのみ有効です。

[kms:GenerateMac](#)

キーユーザーが HMAC KMS キーを使用して HMAC タグを生成できるようにします。

[kms:GetPublicKey](#)

非対称 KMS キーのパブリックキーをキーユーザーがダウンロードできるようにします。このパブリックキーを共有する関係者は、外部でデータを暗号化できます AWS KMS。ただし、これらの暗号テキストは、AWS KMSで [Decrypt](#) オペレーションを呼び出すことによってのみ復号できます。

[kms:ReEncrypt*](#)

この KMS キーで最初に暗号化されたデータの再暗号化、またはこの KMS キーを使用して以前に暗号化されたデータの再暗号化をキーユーザーが行えるようにします。[ReEncrypt](#) オペレーションでは、ソースおよび送信先 KMS キーの両方にアクセスする必要があります。これを行うには、ソース KMS キーの `kms:ReEncryptFrom` アクセス許可と宛先 KMS キーの `kms:ReEncryptTo` アクセス許可を許可します。ただし、わかりやすいようにコンソールでは、両方の KMS キーで `kms:ReEncrypt*` を (* をワイルドカード文字で) 許可します。

[kms:Sign](#)

この KMS キーでメッセージにキーユーザーが署名できるようにします。

[kms:Verify](#)

この KMS キーで署名をキーユーザーが検証できるようにします。

[kms:VerifyMac](#)

キーユーザーが HMAC KMS キーを使用して HMAC タグを検証できるようにします。

AWS サービスで KMS キーを使用することをキーユーザーに許可する

コンソールのデフォルトのキーポリシーでは、`grants: AWS services` を使用する AWS サービスでデータを保護するために必要な権限をキーユーザーに付与します。多くの場合、権限を使用して、KMS キーを使用するための具体的で制限された権限を取得します。

このキーポリシーステートメントは、キーユーザーが KMS キーに対する許可を作成、表示、および取り消すことを許可します。ただし、許可オペレーションリクエストが [AWS 統合された サービス AWS KMS](#) から送信された場合に限り、[kms:GrantIsForAWSResource](#) ポリシー条件では、ユーザーがこれらの許可オペレーションを直接呼び出すことはできません。キーユーザーが許可すると、AWS サービスはユーザーに代わって許可を作成し、KMS キーを使用してユーザーのデータを保護することをサービスに許可できます。

キーユーザーが統合サービスで KMS キーを使用するには、これらのグラント許可が必要になりますが、これらの許可だけでは不十分です。キーユーザーには、統合されたサービスを使用するアクセス

許可も必要です。と統合する AWS サービスへのアクセス権をユーザーに付与する方法の詳細については AWS KMS、統合サービスのドキュメントを参照してください。

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/ExampleKeyUserRole"},
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

例えば、キーユーザーは以下の方法により、KMS キーでこれらのアクセス許可を使用できます。

- この KMS キーを Amazon Elastic Block Store (Amazon EBS) および Amazon Elastic Compute Cloud (Amazon EC2) とともに使用して、暗号化された EBS ボリュームを EC2 インスタンスにアタッチします。キーユーザーは、KMS キーを使用して暗号化されたボリュームをインスタンスにアタッチするアクセス許可を、Amazon EC2 に暗黙的に付与します。詳細については、「[Amazon Elastic Block Store \(Amazon EBS\) が 使用方法 AWS KMS](#)」を参照してください。
- この KMS キーを Amazon Redshift とともに使用して、暗号化されたクラスターを起動します。キーユーザーは、KMS キーを使用して暗号化されたクラスターを起動し、暗号化されたスナップショットを作成するアクセス許可を、Amazon Redshift に暗黙的に付与します。詳細については、「[Amazon Redshift が 使用方法 AWS KMS](#)」を参照してください。
- この KMS キーを、[AWS KMSと統合された他のAWS サービス](#)で使します。これらのサービスは、暗号化されたリソースを作成、管理したり、これらのサービスで使用する際にグラントを使します。

デフォルトキーポリシーにより、キーユーザーは、自身のグラント許可をグラントを使用するすべての統合サービスに付与することができます。ただし、指定された AWS サービスへのアクセス許可を制限するカスタムキーポリシーを作成できます。詳細については、「[kms:ViaService](#)」の条件キーを参照してください。

キーポリシーを表示する

[AWS マネージドキー](#) アカウントの AWS KMS [カスタマーマネージドキー](#) または [のキーポリシー](#) は、AWS KMS コンソールまたは AWS KMS API の [GetKeyPolicy](#) オペレーションを使用して表示できます。これらの手法を使用して別の AWS アカウントにある KMS キーのキーポリシーを表示することはできません。

AWS KMS キーポリシーの詳細については、「」を参照してください。[のキーポリシー AWS KMS](#)。KMS キーにアクセスできるユーザーとロールを特定する方法については、[the section called “アクセスの確認”](#) を参照してください。

AWS KMS コンソールの使用

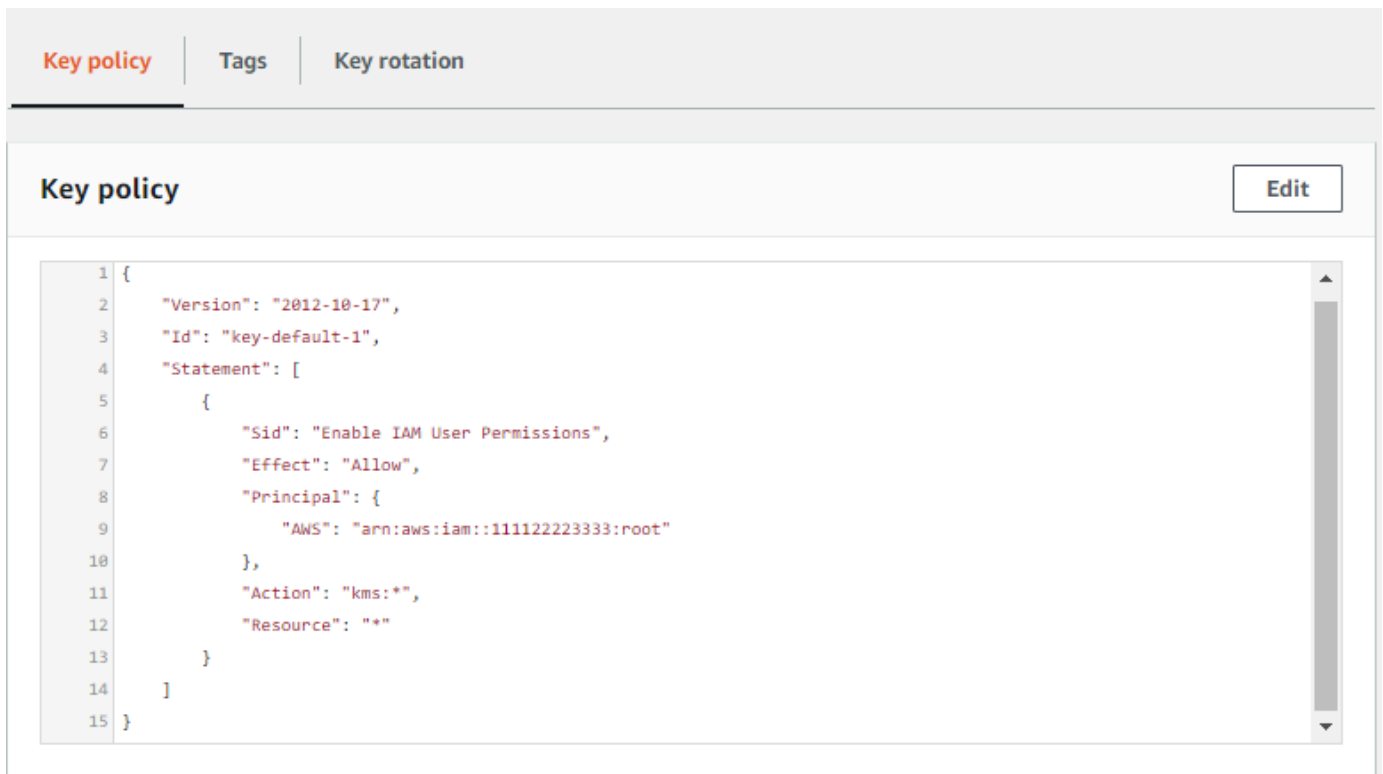
認可されたユーザーは、AWS Management Console の [Key policy] (キーポリシー) タブで、[AWS マネージドキー](#) または [カスタマーマネージドキー](#) のキーポリシーを表示できます。

で KMS キーのキーポリシーを表示するには AWS Management Console、[kms:ListAliases](#)、[kms:DescribeKey](#)、および [kms:GetKeyPolicy](#) アクセス許可が必要です。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. が AWS 作成および管理するアカウントのキーを表示するには、ナビゲーションペインでマネージドAWS キーを選択します。ユーザーが作成および管理するアカウント内のキーを表示するには、ナビゲーションペインで [Customer managed keys] (カスタマーマネージドキー) を選択します。
4. KMS キーのリストで、確認する KMS キーのエイリアスまたはキー ID を選択します。
5. [Key policy] (キーポリシー) タブを選択します。

[キーポリシー] タブに、キーポリシードキュメントが表示されることがあります。これはポリシービューです。キーポリシーステートメントでは、キーポリシーによって KMS キーへのアクセス許可を付与されたプリンシパルを表示して、それらが実行できるアクションを確認できます。

次の例は、[デフォルトのキーポリシー](#) のポリシービューを示しています。



または、で KMS キーを作成した場合 AWS Management Console、キー管理者、キー削除、キーユーザーのセクションを含むデフォルトビューが表示されます。キーポリシードキュメントを表示するには、[ポリシービューに切り替える] を選択します。

次の例は、[デフォルトのキーポリシー](#)のデフォルトのビューを示しています。

Key policy

Tags

Key rotation

Key policy

Switch to policy view

Key administrators

Choose the IAM users and roles who can administer this key through the KMS API. You might need to add additional permissions for the users or roles to administer this key from this console. [Learn more](#)

Add

Remove

Q

< 1 >

	Name	Path	Type
Empty Resources			
No resources to display			

Key deletion

☐ Allow key administrators to delete this key

Key users

The following IAM users and roles can use this key to encrypt and decrypt data from within applications and when using AWS services integrated with KMS. [Learn more](#)

Add

Remove

Q

< 1 >

	Name	Path	Type
Empty Resources			
No resources to display			

AWS KMS API の使用

で KMS キーのキーポリシーを取得するには AWS アカウント、API の [GetKeyPolicy](#) AWS KMS オペレーションを使用します。このオペレーションを使用して、別のアカウントのキーポリシーを表示することはできません。

次の例では、AWS Command Line Interface (AWS CLI) で [get-key-policy](#) コマンドを使用しますが、任意の AWS SDK を使用してこのリクエストを行うことができます。

キーポリシーを表示する

99

default が唯一の有効な値であっても、PolicyName パラメータ は必須であることに注意してください。また、このコマンドは、表示を容易にするために、JSON ではなくテキストでの出力を要求します。

このコマンドを実行する前に、サンプルキー ID をアカウントの有効なキー ID に置き換えます。

```
$ aws kms get-key-policy --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --policy-name default --output text
```

レスポンスは、[デフォルトのキーポリシー](#)を返す、次のようなものである必要があります。

JSON

```
{
  "Version" : "2012-10-17",
  "Id" : "key-consolepolicy-3",
  "Statement" : [ {
    "Sid" : "EnableIAMUserPermissions",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

キーポリシーを変更する

AWS Management Console または [PutKeyPolicy](#) オペレーション AWS アカウント を使用して、の KMS キーのキーポリシーを変更できます。これらの手法を使用して別の AWS アカウントにある KMS キーのキーポリシーを変更することはできません。

キーポリシーを変更する場合は、以下のルールに注意してください。

- [AWS マネージドキー](#) または [カスターマネージドキー](#) のキーポリシーを表示できますが、変更できるのは、カスターマネージドキーのキーポリシーのみです。の AWS マネージドキーポリシーは、アカウントで KMS キーを作成した AWS サービスによって作成および管理されます。[AWS 所有のキー](#) のキーポリシーは表示または変更できません。

- キーポリシー AWS アカウント で IAM ユーザー、IAM ロール、および を追加または削除し、それらのプリンシパルに対して許可または拒否されるアクションを変更できます。キーポリシーでプリンシパルとアクセス権限を指定する方法については、「[キーポリシー](#)」を参照してください。
- IAM グループをキーポリシーに追加することはできませんが、複数の IAM ユーザーおよび IAM ロールを追加できます。詳細については、「[複数の IAM プリンシパルが KMS キーにアクセスできるようにする](#)」を参照してください。
- キーポリシー AWS アカウント に外部を追加する場合は、外部アカウントの IAM ポリシーを使用して、それらのアカウントの IAM ユーザー、グループ、またはロールにアクセス許可を付与する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。
- 結果として得られるキーポリシードキュメントは 32 KB (32,768 バイト) を超えることはできません。

キーポリシーを変更する方法

キーポリシーは、以下のセクションで説明している 3 つの異なる方法で変更できます。

トピック

- [AWS Management Console のデフォルトビューの使用](#)
- [AWS Management Console ポリシービューの使用](#)
- [AWS KMS API の使用](#)

AWS Management Console のデフォルトビューの使用

コンソールを使用して、デフォルトビューと呼ばれるグラフィカルインターフェイスで、キーポリシーを変更できます。

以下のステップがコンソールの表示と一致しない場合、このキーポリシーはコンソールで作成されなかった可能性があります。または、コンソールのデフォルトビューがサポートしていない方法でキーポリシーが変更されている可能性があります。その場合は、「[AWS Management Console ポリシービューの使用](#)」または「[AWS KMS API の使用](#)」の手順に従ってください。

1. [AWS KMS コンソールの使用](#) の説明に従って、カスターマネージドキーのキーポリシーを表示します。(のキーポリシーは変更できません) AWS マネージドキー。
2. 変更する対象を決定します。

- [キー管理者](#)を追加または削除し、キー管理者による [KMS キーの削除](#)を許可または拒否するには、ページの [Key administrators] (キー管理者)セクションのコントロールを使用します。キー管理者は、KMS キーの有効化と無効化、キーポリシーの設定、[キーローテーションの有効化](#)などを含む KMS キーの管理を行います。
- [キーユーザー](#)を追加または削除し、外部に KMS キー AWS アカウント の使用を許可または禁止するには、ページのキーユーザーセクションのコントロールを使用します。キーユーザーは、データキーの暗号化、復号、再暗号化、生成などの[暗号化オペレーション](#)で KMS キーを使用できます。

AWS Management Console ポリシービューの使用

コンソールのポリシービューで、キーポリシードキュメントを変更できます。

1. [AWS KMS コンソールの使用](#) の説明に従って、カスタマーマネージドキーのキーポリシーを表示します。(のキーポリシーは変更できません) AWS マネージドキー。
2. [Key Policy (キーポリシー)] セクションで、[Switch to policy view (ポリシービューへの切り替え)] を選択します。
3. [Edit] を選択します。
4. 変更する対象を決定します。
 - 新しいステートメントを追加するには、新しいステートメントの追加を選択します。次に、ステートメントビルダーパネルにリストされているオプションから新しいキーポリシーステートメントのアクション、プリンシパル、および条件を選択するか、ポリシーステートメント要素を手動で入力できます。
 - キーポリシーからステートメントを削除するには、ステートメントを選択し、削除を選択します。選択したポリシーステートメントを確認し、削除することを確認します。ステートメントの削除を続行しない場合は、キャンセルを選択します。
 - 既存のキーポリシーステートメントを編集するには、ステートメントを選択します。次に、ステートメントビルダーパネルを使用して、変更する特定の要素を選択するか、ステートメントを手動で編集できます。
5. [Save changes] (変更の保存) をクリックします。

AWS KMS API の使用

[PutKeyPolicy](#) オペレーションを使用して、の KMS キーのキーポリシーを変更できます AWS アカウント。この API を別の AWS アカウントの KMS キーで使用することはできません。

1. [GetKeyPolicy](#) オペレーションを使用して、既存のキーポリシードキュメントを取得し、そのキーポリシードキュメントをファイルに保存します。複数のプログラミング言語のサンプルコードについては、「[AWS SDK または CLI GetKeyPolicyで 使用する](#)」を参照してください。
2. 任意のテキストエディタでキーポリシードキュメントを開き、キーポリシードキュメントを編集してファイルを保存します。
3. [PutKeyPolicy](#) オペレーションを使用して、更新されたキーポリシードキュメントを KMS キーに適用します。複数のプログラミング言語のサンプルコードについては、「[AWS SDK または CLI PutKeyPolicyで 使用する](#)」を参照してください。

ある KMS キーから別の KMS キーにキーポリシーをコピーする例については、AWS CLI コマンドリファレンスの [GetKeyPolicy の例](#)を参照してください。

キーポリシーでの AWS サービスのアクセス許可

多くの AWS サービスは AWS KMS keys、を使用して管理するリソースを保護します。あるサービスで [AWS 所有のキー](#) または [AWS マネージドキー](#) が使用される場合、そのサービスではこれらの KMS キーのキーポリシーが確立されて管理されます。

ただし、お客様が AWS のサービスで [カスターマネージドキー](#)を使用する場合は、ご自身でキーポリシーを設定して管理します。ユーザーに代わってリソースを保護するのに必要な最小限のアクセス許可が、そのキーポリシーによってサービスに付与される必要があります。最小特権の原則 (サービスに必要なアクセス許可のみを付与) に従うことをお勧めします。どのアクセス許可がそのサービスで必要かを把握し、[AWS グローバル条件キー](#)および [AWS KMS 条件キー](#)を使用してアクセス許可を絞り込むことで、これを効果的に行うことができます。

カスターマネージドキーに対してサービスで必要になるアクセス許可を調べるには、そのサービスの暗号化に関するドキュメントを参照してください。例えば、Amazon Elastic Block Store (Amazon EBS) で必要なアクセス許可については、「[Amazon EC2 ユーザーガイド](#)」の「IAM ユーザー向けアクセス許可」および「[Amazon EC2 ユーザーガイド](#)」を参照してください。Secrets Manager で必要なアクセス許可については、AWS Secrets Manager ユーザーガイドの [Authorizing use of the KMS key](#) を参照してください。

での IAM ポリシーの使用 AWS KMS

IAM ポリシーとキー[ポリシー](#)、[権限](#)、[VPC エンドポイントポリシー](#)を使用して、AWS KMS keys へのアクセスを制御できます AWS KMS。

Note

IAM ポリシーを使用して KMS キーへのアクセスを制御するには、KMS キーのキーポリシーが IAM ポリシーを使用するアクセス許可をアカウントに付与する必要があります。具体的には、キーポリシーには [IAM ポリシーを有効にするポリシーステートメント](#) を含める必要があります。

このセクションでは、IAM ポリシーを使用して AWS KMS オペレーションへのアクセスを制御する方法について説明します。IAM の一般的な情報については、「[IAM ユーザーガイド](#)」を参照してください。

すべての KMS キーはキーポリシーを持つ必要があります。IAM ポリシーはオプションです。IAM ポリシーを使用して KMS キーへのアクセスを制御するには、KMS キーのキーポリシーが IAM ポリシーを使用するアクセス許可をアカウントに付与する必要があります。具体的には、キーポリシーには [IAM ポリシーを有効にするポリシーステートメント](#) を含める必要があります。

IAM ポリシーは、任意の AWS KMS オペレーションへのアクセスを制御できます。キーポリシーとは異なり、IAM ポリシーは複数の KMS キーへのアクセスを制御し、いくつかの関連 AWS サービスのオペレーションに対するアクセス許可を提供できます。IAM ポリシーは、特定の KMS キーを含まないため、キーポリシーで制御できないオペレーション ([CreateKey](#) など) へのアクセスを制御する場合に特に便利です。

Amazon Virtual Private Cloud (Amazon VPC) エンドポイント AWS KMS を介して にアクセスする場合は、VPC エンドポイントポリシーを使用して、エンドポイントの使用時に AWS KMS リソースへのアクセスを制限することもできます。たとえば、VPC エンドポイントを使用する場合、 のプリンシパルのみがカスタマーマネージドキー AWS アカウント にアクセスすることを許可できます。詳細については、「[VPC エンドポイントポリシー](#)」を参照してください。

JSON ポリシードキュメントの記述と書式設定については、『[IAM ユーザーガイド](#)』の「[IAM JSON ポリシーリファレンス](#)」を参照してください。

IAM ポリシーは、次の方法で使用できます。

- フェデレーションまたはクロスアカウントアクセス権限のロールにアクセス許可ポリシーをアタッチする — IAM ロールに IAM ポリシーをアタッチして、ID フェデレーションを有効にしたり、クロスアカウントアクセス権限を許可したり、EC2 インスタンスで実行されているアプリケーションにアクセス許可を付与したりできます。IAM ロールのさまざまなユースケースの詳細については、IAM ユーザーガイドの [IAM ロール](#) を参照してください。
- ユーザーまたはグループにアクセス許可ポリシーをアタッチする — ユーザーまたはユーザーのグループに AWS KMS オペレーションの呼び出しを許可するポリシーをアタッチできます。ただし、IAM ベストプラクティスでは、可能な限り IAM ロールなどの一時的な認証情報を持つアイデンティティを使用することが推奨されています。

次の例は、アクセス AWS KMS 許可を持つ IAM ポリシーを示しています。このポリシーは、アタッチされた IAM アイデンティティで、すべての KMS キーとエイリアスを一覧表示できるようにします。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

すべての IAM ポリシーと同様に、このポリシーには Principal 要素がありません。IAM アイデンティティに IAM ポリシーをアタッチすると、そのアイデンティティは、ポリシーで指定されたアクセス権限を取得します。

すべての AWS KMS API アクションとそれらが適用されるリソースを示す表については、「」を参照してください [アクセス許可に関するリファレンス](#)。

複数の IAM プリンシパルが KMS キーにアクセスできるようにする

IAM グループは、キーポリシー内の有効なプリンシパルではありません。複数のユーザーおよびロールが KMS キーにアクセスできるようにするには、次のいずれかを行います。

- IAM ロールをキーポリシーのプリンシパルとして使用します。必要に応じて、複数の権限を持つユーザーがそのロールを引き受けることができます。詳細については、「IAM ユーザーガイド」の「[IAM ロール](#)」を参照してください。

複数の IAM ユーザーをキーポリシーにリストすることは可能ですが、許可されたユーザーのリストが変更されるたびにキーポリシーを更新する必要があるため、この方法は推奨されません。また、IAM のベストプラクティスでは、長期的な認証情報を持つ IAM ユーザーの使用は推奨されていません。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

- IAM ポリシーを使用して IAM グループに許可を付与します。そのためには、キーポリシーに、[IAM ポリシーで KMS キーへのアクセスを許可する](#)ステートメントが含まれていることを確認します。次に、KMS キーへのアクセスを許可する [IAM ポリシーを作成し](#)、そのポリシーを [IAM グループ \(許可された IAM ユーザーを含む\) にアタッチします](#)。このアプローチを使用すると、承認されたユーザーのリストが変更されたときにポリシーを変更する必要はありません。代わりに、適切な IAM グループに対してそれらのユーザーを追加または削除するだけで済みます。詳細については、「IAM ユーザーガイド」の「[IAM ユーザーグループ](#)」を参照してください。

AWS KMS キーポリシーと IAM ポリシーの連携の詳細については、「」を参照してください [アクセス AWS KMS 許可のトラブルシューティング](#)。

IAM ポリシーのベストプラクティス

へのアクセスを保護する AWS KMS keys ことは、すべての AWS リソースのセキュリティにとって重要です。KMS キーは、AWS アカウントで最も機密性の高いリソースの多くを保護するために使用されます。KMS キーへのアクセスを制御する [キーポリシー](#)、IAM ポリシー、[グラント](#) (権限付与)、VPC エンドポイントポリシーを設計します。

KMS キーへのアクセスを制御する IAM ポリシーステートメントでは、[最小権限の原則](#)を使用します。IAM プリンシパルで、使用または管理する必要がある KMS キーのみに対して、必要なアクセス許可のみを付与します。

以下のベストプラクティスは、AWS KMS キーとエイリアスへのアクセスを制御する IAM ポリシーに適用されます。IAM ポリシーの一般的なベストプラクティスのガイダンスについては、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

キーポリシーを使用する

可能であれば、他の AWS アカウントを含む多くの KMS キーに適用できる IAM ポリシーではなく、1 つの KMS キーに影響するキーポリシーでアクセス許可を付与します。これは、[kms:PutKeyPolicy](#) や [kms:ScheduleKeyDeletion](#) などの機密性の高いアクセス許可にとって特に重要ですが、データの保護方法を決定する暗号化オペレーションでも重要です。

CreateKey アクセス許可を制限する

キー ([kms:CreateKey](#)) を作成する権限を必要なプリンシパルにのみ付与します。KMS キーを作成するプリンシパルは、そのキーポリシーも設定するため、自分自身や他のユーザーに、作成した KMS キーを使用および管理するためのアクセス許可を付与できます。このアクセス許可を許可する場合は、[ポリシー条件](#)を使用して制限することを検討してください。例えば、[kms:KeySpec](#) 条件を使用して、対称暗号化 KMS キーに対する許可を制限できます。

IAM ポリシーで KMS キーを指定する

ベストプラクティスとして、ポリシーステートメントの Resource 要素でアクセス許可が適用する各 KMS キーの [キー ARN](#) を指定します。この方法は、プリンシパルが必要とする KMS キーへのアクセス許可を制限します。例えば、この Resource 要素は、プリンシパルが使用する必要がある KMS キーのみを一覧表示します。

```
"Resource": [  
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
    "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"  
]
```

KMS キーを指定するのが現実的でない場合は、などの信頼できる AWS アカウント およびリージョンの KMS キーへのアクセスを制限する Resource 値を使用します `arn:aws:kms:region:account:key/*`。または、など AWS アカウント、信頼された のすべてのリージョン (*) の KMS キーへのアクセスを制限します `arn:aws:kms:*:account:key/*`。

[キー ID](#)、[エイリアス名](#)、[エイリアス ARN](#) を使用して、IAM ポリシーの Resource フィールド内の KMS キーを表すことはできません。エイリアス ARN を指定する場合、ポリシーは KMS キーではなく、エイリアスに適用されます。エイリアスの IAM ポリシーの詳細については、[エイリアスへのアクセスの制御](#) を参照してください。

IAM ポリシーの「リソース」:「*」を避ける

ワイルドカード文字 (*) を慎重に使用してください。キーポリシーでは、Resource 要素のワイルドカード文字は、キーポリシーがアタッチされている KMS キーを表します。ただし、IAM ポリシーでは、Resource 要素 ("Resource": "*") のワイルドカード文字だけが、プリンシパルのアカウント AWS アカウント が使用するアクセス許可を持つすべての KMS キーにアクセス許可を適用します。これには、[他の の KMS キー AWS アカウント](#)と、プリンシパルのアカウントの KMS キーが含まれる場合があります。

たとえば、別の で KMS キーを使用するには AWS アカウント、プリンシパルに、外部アカウントの KMS キーのキーポリシーと、自分のアカウントの IAM ポリシーからのアクセス許可が必要です。任意のアカウントが自分の KMS キーに対する AWS アカウント [kms:Decrypt](#) アクセス許可を付与したとします。この場合、すべての KMS キー ("Resource": "*") に対するロール [kms:Decrypt](#) のアクセス許可を付与するアカウントの IAM ポリシーは、要件の IAM パートを満たします。その結果、そのロールを引き受けることができるプリンシパルは、信頼されていないアカウントの KMS キーを使用して、暗号テキストを復号できるようになります。オペレーションのエントリは、両方のアカウントの CloudTrail ログに表示されます。

特に、次の API オペレーションを許可するポリシーステートメントでは使用しないでください。"Resource": "*"これらのオペレーションは、他の の KMS キーで呼び出すことができます AWS アカウント。

- [DescribeKey](#)
- [GetKeyRotationStatus](#)
- [暗号化オペレーション](#)
([Encrypt](#)、[Decrypt](#)、[GenerateDataKey](#)、[GenerateDataKeyPair](#)、[GenerateDataKeyWithoutPlaintext](#)、[CreateGrant](#)、[ListGrants](#)、[ListRetirableGrants](#)、[RetireGrant](#)、[RevokeGrant](#))

「リソース」を使用する場合:「*」

IAM ポリシーでは、Resource 要素でワイルドカード文字は、それを必要とするアクセス権限に対してのみ使用します。"Resource": "*" 要素が必要なのは、次の権限のみです。

- [KMS: 作成キー](#)
- [KMS: ジェネラタランダム](#)
- [kms:ListAliases](#)
- [KMS: リストキー](#)
- [kms:CreateCustomKeyStore](#) や [kms:ConnectCustomKeyStore](#) などのカスタムキーストアのアクセス許可。などのカスタムキーストアの権限。

Note

エイリアスオペレーションのアクセス許可

([kms:CreateAlias](#)、[kms:UpdateAlias](#)、[kms>DeleteAlias](#)) は、エイリアスと KMS キーにタッチする必要があります。IAM ポリシーで "Resource": "*" を使用し、エイリアスと KMS キーを表すことも、Resource エlement でエイリアスと KMS キーを指定することもできます。例については「[エイリアスへのアクセスの制御](#)」を参照してください。

このトピックの例では、KMS キーの IAM ポリシーを設計するための詳細情報とガイダンスを示します。すべての AWS リソースの IAM ベストプラクティスについては、IAM [ユーザーガイドの「IAM のセキュリティのベストプラクティス」](#)を参照してください。

IAM ポリシーステートメントで KMS キーを指定する

IAM ポリシーを使用して、プリンシパルに KMS キーの使用または管理を許可できます。KMS キーは、ポリシーステートメントの Resource 要素で指定されます。

- IAM ポリシーステートメントで KMS キーを指定するには、[キー ARN](#) を使用する必要があります。[キー ID](#)、[エイリアス名](#)、[エイリアス ARN](#) を使用して IAM ポリシーステートメントの KMS キーを識別することはできません。

例: 「Resource»: "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"」

エイリアスに基づいて KMS キーへのアクセスを制御するには、[kms:RequestAlias](#) または [kms:ResourceAliases](#) 条件キーを使用します。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

[CreateAlias](#)、[UpdateAlias](#)、[DeleteAlias](#) などのエイリアスオペレーションへのアクセスを制御する、ポリシーステートメント内のみのリソースとしてエイリアス ARN を使用します。 詳細については、「[エイリアスへのアクセスの制御](#)」を参照してください。

- アカウントとリージョンで複数の KMS キーを指定するには、キー ARN のリージョンまたはリソース ID の位置にワイルドカード文字 (*) を使用します。

例えば、アカウントの米国西部 (オレゴン) リージョンのすべての KMS キーを指定するには、「Resource»: "arn:aws:kms:us-west-2:111122223333:key/*" を使用しま

す。アカウントのすべてのリージョンですべての KMS キーを指定するには、「Resource」:
"arn:aws:kms:*:111122223333:key/*" を使用します。

- すべての KMS キーを表すには、ワイルドカード文字のみ ("*") を使用します。この形式は、特定の KMS キーを使用しないオペレーション、つまり [CreateKey](#)、[GenerateRandom](#)、[ListAliases](#)、[ListKeys](#) などに使用します。

ポリシーステートメントを書き込むときは、すべての KMS キーへのアクセス許可を付与するのではなく、プリンシパルが使用する必要がある KMS キーだけを指定するのが [ベストプラクティス](#) です。

例えば、次の IAM ポリシーステートメントで、プリンシパルは、ポリシーステートメントの Resource 要素に記載されている KMS キーに対してのみ、[DescribeKey](#)、[GenerateDataKey](#)、[Decrypt](#) オペレーションを呼び出すことができます。キー ARN による KMS キーの指定は、アクセス許可を指定した KMS キーのみに制限するベストプラクティスです。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": [
      "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "arn:aws:kms:us-west-2:111122223333:key/01234abcd-12ab-34cd-56ef-1234567890ab"
    ]
  }
}
```

特定の信頼された のすべての KMS キーにアクセス許可を適用するには AWS アカウント、リージョンとキー ID の位置でワイルドカード文字 (*) を使用できます。例えば、次のポリシーステートメン

トでは、プリンシパルが 2 つの信頼できるサンプルアカウントのすべての KMS キーで、指定されたオペレーションを呼び出すことができます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:GenerateDataKeyPair"
    ],
    "Resource": [
      "arn:aws:kms:*:111122223333:key/*",
      "arn:aws:kms:*:444455556666:key/*"
    ]
  }
}
```

Resource 要素内でワイルドカード文字 ("*") を単独で使用することもできます。アカウントが使用を許可されているすべての KMS キーへのアクセスを許可するため、主に、特定の KMS キーおよび Deny ステートメントを含まないオペレーションに推奨されます。また、機密性の低い読み取り専用オペレーションのみを許可するポリシーステートメントで使用することもできます。AWS KMS オペレーションに特定の KMS キーが含まれているかどうかを判断するには、[the section called “アクセス許可に関するリファレンス”](#)。

例えば、次のポリシーステートメントでは、Deny エフェクトを使用して、プリンシパルが任意の KMS キーに対して指定されたオペレーションを使用できないようにします。Resource 要素でワイルドカード文字を使用して、すべての KMS キーを表します。

JSON

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

```
    "kms:CreateKey",
    "kms:PutKeyPolicy",
    "kms:CreateGrant",
    "kms:ScheduleKeyDeletion"
  ],
  "Resource": "*"
}
```

次のポリシーステートメントでは、ワイルドカード文字だけを使用してすべての KMS キーを表します。ただし、機密性の低い読み取り専用オペレーションと、特定の KMS キーに適用されないオペレーションのみを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:CreateKey",
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:ListResourceTags"
    ],
    "Resource": "*"
  }
}
```

IAM ポリシーの例

このセクションでは、さまざまな AWS KMS アクションのためのアクセス許可を付与する IAM ポリシーの例を示しています。

⚠ Important

以下のポリシーのアクセス許可の一部は、KMS キーのキーポリシーも同じアクセス許可を付与する場合にのみ付与されます。詳細については、「[アクセス許可に関するリファレンス](#)」を参照してください。

JSON ポリシードキュメントの記述と書式設定については、『[IAM ユーザーガイド](#)』の「[IAM JSON ポリシーリファレンス](#)」を参照してください。

例

- [AWS KMS コンソールで KMS キーの表示をユーザーに許可する](#)
- [KMS キーの作成をユーザーに許可する](#)
- [特定の の任意の KMS キーを使用して暗号化および復号化することをユーザーに許可する AWS アカウント](#)
- [特定の およびリージョン内の任意の KMS キーを使用して暗号化 AWS アカウント および復号化することをユーザーに許可する](#)
- [特定の KMS キーにより暗号化および復号化することをユーザーに許可する](#)
- [KMS キーの無効化または削除を禁止する](#)

AWS KMS コンソールで KMS キーの表示をユーザーに許可する

次の IAM ポリシーでは、AWS KMS コンソールへの読み取り専用アクセスをユーザーに許可します。これらのアクセス許可を持つユーザーは、内のすべての KMS キーを表示できますが AWS アカウント、KMS キーを作成または変更することはできません。

AWS マネージドキー および [Customer managed keys] (カスタマーマネージドキー) ページ上に KMS キーを表示するには、キーにタグやエイリアスがない場合でも、プリンシパルに [kms:ListKeys](#)、[kms:ListAliases](#)、[tag:GetResources](#) への許可が必要です。KMS キーの詳細ページでオプションの KMS キーテーブルの列とデータを表示するには、残りのアクセス許可 (特に [kms:DescribeKey](#)) が必要です。エラーなしでデフォルトのビューでキーポリシーを表示するには、[iam:ListUsers](#) および [iam:ListRoles](#) アクセス許可が必要です。カスタムキーストアページのデータおよびカスタムキーストアの KMS キーの詳細を表示するには、プリンシパルに [kms:DescribeCustomKeyStores](#) アクセス許可も必要です。

ユーザーのコンソールアクセスを特定の KMS キーに制限すると、コンソールは、表示されない各 KMS キーにエラーを表示します。

このポリシーには 2 つのポリシーステートメントが含まれます。最初のポリシーステートメントの Resource 要素は、例 AWS アカウントのすべてのリージョンのすべての KMS キーで指定されたアクセス許可を付与します。AWS KMS コンソールはプリンシパルのアカウントの KMS キーのみを表示するため、コンソールビューワーは追加のアクセスを必要としません。これは、他ので KMS キーを表示するアクセス許可がある場合にも当てはまります AWS アカウント。残りの AWS KMS および IAM アクセス許可には、特定の KMS キーに適用されないため、"Resource": "*" 要素が必要です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnlyAccessForAllKMSKeysInAccount",
      "Effect": "Allow",
      "Action": [
        "kms:GetPublicKey",
        "kms:GetKeyRotationStatus",
        "kms:GetKeyPolicy",
        "kms:DescribeKey",
        "kms:ListKeyPolicies",
        "kms:ListResourceTags",
        "tag:GetResources"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "ReadOnlyAccessForOperationsWithNoKMSKey",
      "Effect": "Allow",
      "Action": [
        "kms:ListKeys",
        "kms:ListAliases",
        "iam:ListRoles",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

KMS キーの作成をユーザーに許可する

以下の IAM ポリシーは、ユーザーがすべてのタイプの KMS キーを作成することを許可します。Resource 要素の値は `*` です。これは、CreateKey オペレーションが特定の AWS KMS リソース (KMS キーまたはエイリアス) を使用しないためです。

ユーザーを特定タイプの KMS キーに制限するには、[kms:KeySpec](#)、[kms:KeyUsage](#)、および [kms:KeyOrigin](#) 条件キーを使用します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "kms:CreateKey",
    "Resource": "*"
  }
}
```

キーを作成するプリンシパルには、関連するパーミッションが必要な場合があります。

- `kms:PutKeyPolicy` — `kms:CreateKey` アクセス許可を持つプリンシパルには、KMS キーの初期キーポリシーを設定できます。ただし、CreateKey 発信者には、KMS キーのキーポリシーを変更できる [kms:PutKeyPolicy](#) 許可が必要です。または、CreateKey の `BypassPolicyLockoutSafetyCheck` パラメータを指定する必要があります (非推奨)。CreateKey 発信者は、IAM ポリシーから KMS キーの `kms:PutKeyPolicy` アクセス許可を取得することも、作成している KMS キーのキーポリシーにこのアクセス許可を含めることもできます。
- `kms:TagResource` — CreateKey オペレーション中に KMS キーにタグを追加するには、CreateKey 発信者に IAM ポリシーの [kms:TagResource](#) アクセス許可が必要です。新しい KMS キーのキーポリシーにこのアクセス許可を含めるだけでは不十分です。ただし、CreateKey 発信者が初期キーポリシーの `kms:TagResource` を含む場合は、KMS キーの作成後に、個別の呼び出しでタグを追加できます。
- `kms:CreateAlias` — AWS KMS コンソールで KMS キーを作成するプリンシパルには、KMS キーとエイリアスに対する [kms:CreateAlias](#) アクセス許可が必要です。(コンソールは、へのコール CreateKey とへのコールを 2 つ行います CreateAlias)。IAM ポリシーでエイリアスアクセス

許可を指定する必要があります。キーポリシーまたは IAM ポリシーで、KMS キーアクセス許可を付与できます。詳細については、「[エイリアスへのアクセスの制御](#)」を参照してください。

に加えて `kms:CreateKey`、次の IAM ポリシーは、内のすべての KMS キーに対する `kms:TagResource` アクセス許可 AWS アカウント と、アカウント内のすべてのエイリアスに対する `kms:CreateAlias` アクセス許可を提供します。また、IAM ポリシーでのみ提供できる便利な読み取り専用アクセス許可もいくつか含まれています。

この IAM ポリシーには、キーポリシーで設定できる `kms:PutKeyPolicy` アクセス権限やその他のアクセス権限は含まれていません。これらのアクセス許可は、1 つの KMS キーにのみ適用されるキーポリシーで設定するのが [ベストプラクティス](#) です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPermissionsForParticularKMSKeys",
      "Effect": "Allow",
      "Action": "kms:TagResource",
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPermissionsForParticularAliases",
      "Effect": "Allow",
      "Action": "kms:CreateAlias",
      "Resource": "arn:aws:kms:*:111122223333:alias/*"
    },
    {
      "Sid": "IAMPermissionsForAllKMSKeys",
      "Effect": "Allow",
      "Action": [
        "kms:CreateKey",
        "kms:ListKeys",
        "kms:ListAliases"
      ],
      "Resource": "*"
    }
  ]
}
```

```
}
```

特定の の任意の KMS キーを使用して暗号化および復号化することをユーザーに許可する AWS アカウント

次の IAM ポリシーでは、 の任意の KMS キーを使用してデータを暗号化および復号化できます AWS アカウント 111122223333。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/*"
  }
}
```

特定の およびリージョン内の任意の KMS キーを使用して暗号化 AWS アカウント および復号化することをユーザーに許可する

次の IAM ポリシーでは、ユーザーは米国西部 (オレゴン) リージョンの で AWS アカウント 111122223333 任意の KMS キーを使用してデータを暗号化および復号化できます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],

```



```
"Resource": [  
  "arn:aws:kms:us-west-2:111122223333:key/*"  
]  
}  
}
```

特定の KMS キーにより暗号化および復号することをユーザーに許可する

次の IAM ポリシーでは、Resource エlement で指定された 2 つの KMS キーを使用して、ユーザーがデータを暗号化および復号できます。IAM ポリシーステートメントで KMS キーを指定するときは、KMS キーの [キー ARN](#) を使用する必要があります。

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Allow",  
    "Action": [  
      "kms:Encrypt",  
      "kms:Decrypt"  
    ],  
    "Resource": [  
      "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
      "arn:aws:kms:us-west-2:111122223333:key/01234abc-d12a-b34c-d56e-f1234567890a"  
    ]  
  }  
}
```

KMS キーの無効化または削除を禁止する

次の IAM ポリシーでは、別の IAM ポリシーまたはキーポリシーで許可されている場合でも、ユーザーが KMS キーを無効化、削除できないようにします。アクセス権限を明示的に拒否するポリシーは、同じアクセス権限を明示的に付与するポリシーを含め、他のすべてのポリシーを上書きします。詳細については、「[アクセス AWS KMS 許可のトラブルシューティング](#)」を参照してください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [
      "kms:DisableKey",
      "kms:ScheduleKeyDeletion"
    ],
    "Resource": "*"
  }
}
```

のリソースコントロールポリシー AWS KMS

リソースコントロールポリシー (RCPs) は、組織内の AWS リソースに予防コントロールを適用するために使用できる組織ポリシーの一種です。RCPsは、AWS リソースへの外部アクセスを大規模に一元的に制限するのに役立ちます。RCPs、サービスコントロールポリシー (SCP)。SCP を使用して組織内の IAM ロールとユーザーに最大アクセス許可を一元的に設定できますが、RCPs を使用して組織内の AWS リソースに最大アクセス許可を一元的に設定できます。

RCPs を使用して、組織内のカスターマネージド KMS キーへのアクセス許可を管理できます。RCPs だけでは、カスターマネージドキーにアクセス許可を付与するには不十分です。RCP によってアクセス許可は付与されません。RCP は、ID が影響を受けるアカウントのリソースに対して実行できるアクションに対するアクセス許可ガードレールを定義するか、制限を設定します。管理者は、実際にアクセス許可を付与するために、IAM ロールまたはユーザー、またはキーポリシーにアイデンティティベースのポリシーをアタッチする必要があります。

Note

組織内のリソースコントロールポリシーは、には適用されません[AWS マネージドキー](#)。
AWS マネージドキー は、ユーザーに代わって AWS サービスによって作成、管理、使用されます。ユーザーのアクセス許可を変更または管理することはできません。

詳細はこちら

- RCPs [「リソースコントロールポリシー」](#)を参照してください。AWS Organizations
- 例を含む RCPs AWS Organizations「ユーザーガイド」の [「RCP 構文」](#)を参照してください。

次の例は、RCP を使用して、外部プリンシパルが組織内のカスタマーマネージドキーにアクセスできないようにする方法を示しています。このポリシーは単なるサンプルであり、独自のビジネスおよびセキュリティニーズに合わせて調整する必要があります。たとえば、ビジネスパートナーによるアクセスを許可するようにポリシーをカスタマイズできます。詳細については、[「データ境界ポリシーの例」リポジトリ](#)を参照してください。

Note

アクセス `kms:RetireGrant` 許可は、Action 要素でアスタリスク (*) をワイルドカードとして指定しても、RCP では有効ではありません。
へのアクセス許可の決定方法の詳細については、`kms:RetireGrant`「」を参照してください [「グラントの使用停止と取り消し」](#)。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RCPEnforceIdentityPerimeter",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "kms:*",
      "Resource": "*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:PrincipalOrgID": "my-org-id"
        },
        "Bool": {
          "aws:PrincipalIsAWSService": "false"
        }
      }
    }
  ]
}
```

次の RCP の例では、リクエストが組織から送信された場合にのみ、AWS サービスプリンシパルがカスタマーマネージド KMS キーにアクセスできることを要求しています。このポリシーは、aws:SourceAccount が存在するリクエストにのみコントロールを適用します。これにより、の使用を必要としないサービス統合aws:SourceAccountには影響しません。aws:SourceAccount がリクエストコンテキストに存在する場合、Null条件は に評価されtrue、aws:SourceOrgIDキーが適用されます。

混乱した代理問題の詳細については、IAM [ユーザーガイドの混乱した代理問題](#)を参照してください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RCPEnforceConfusedDeputyProtection",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "kms:*",
      "Resource": "*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceOrgID": "my-org-id"
        },
        "Bool": {
          "aws:PrincipalIsAWSService": "true"
        },
        "Null": {
          "aws:SourceAccount": "false"
        }
      }
    }
  ]
}
```

の許可 AWS KMS

グラントはポリシーツールであり、[AWS プリンシパル](#)に暗号化オペレーションでの KMS キーの使用を許可します。また、KMS キー (DescribeKey) を表示して、グラントの作成、管理をできるようにします。KMS キーへのアクセスを認可する際、グラントは[キーポリシー](#)および [IAM ポリシー](#)と共に考慮されます。グラントは、作成してそのアクセス許可を使用し、キーポリシーまたは IAM ポリシーを変更することなく削除できるため、一時的なアクセス許可としてよく使用されます。

グラントは、と統合して保管中のデータを暗号化 AWS KMS する AWS サービスで一般的に使用されます。サービスは、アカウント内のユーザーの代わりにグラントを作成し、そのアクセス許可を使用して、タスクが完了するとすぐにグラント廃止にします。AWS サービス、グラントの使用の詳細については、サービスのユーザーガイドまたはデベロッパーガイドの保管時の暗号化トピックを参照してください。

権限は、きわめて柔軟で便利なアクセス制御メカニズムです。KMS キーのグラントを作成すると、このグラントは、グラントで指定されたすべての条件が満たされている場合に限り、被付与者プリンシパルが指定されたグラントオペレーションを KMS キーで呼び出すことを許可します。

- 各グラントでは、正確に 1 つの KMS キーにアクセスできます。異なる AWS アカウントで KMS キーのグラントを作成できます。
- グラントは KMS キーへのアクセスを許可できますが、アクセスを拒否することはできません。
- 各グラントは、それぞれ 1 つの[被付与者プリンシパル](#)を持っています。被付与者プリンシパルは、KMS キー AWS アカウント と同じ または別のアカウントの 1 つ以上の ID を表すことができます。
- グラントは、[グラントオペレーション](#)のみを許可することができます。グラントオペレーションは、グラントの KMS キーによってサポートされている必要があります。サポートされていないオペレーションを指定すると、ValidationError の例外により、[CreateGrant](#) リクエストは失敗します。
- 被付与者プリンシパルは、アクセス許可がキーポリシーまたは IAM ポリシーから付与された場合と同様に、グラントを指定せずに付与されたアクセス許可を使用できます。ただし、AWS KMS API は[結果整合性](#)モデルに従うため、許可を作成、廃止、または取り消すと、変更が全体で利用可能になるまでに少し時間がかかる場合があります AWS KMS。権限でアクセス許可をすぐに使用するには、[権限トークンを使用します](#)。
- 認可されたプリンシパルはグラントを削除できます (グラントの[廃止](#)または[失効](#))。グラントを削除すると、グラントが許可したすべてのアクセス許可が削除されます。グラントを取り消すために追加または削除するポリシーを特定する必要はありません。

- AWS KMS は、各 KMS キーに対する許可の数を制限します。詳細については、「[KMS キーあたりのグラント: 50,000](#)」を参照してください。

グラントを作成するとき、およびグラントを作成する許可を他のユーザーに付与するときは注意が必要です。グラントを作成する許可にはセキュリティ上の影響があります。これは、[kms:PutKeyPolicy](#) 許可でポリシーを設定する場合と似ています。

- KMS キー (`kms:CreateGrant`) の権限を作成する権限を持つユーザーは、権限を使用して、AWS サービスを含むユーザーとロールに KMS キーの使用を許可できます。プリンシパルは、独自の ID AWS アカウント でも、別のアカウントまたは組織の ID でもかまいません。
- 許可は、AWS KMS オペレーションのサブセットのみを許可できます。グラントを使用して、プリンシパルに KMS キーの表示、暗号化オペレーションでの使用、グラントの作成、グラントの廃止を許可できます。詳細については、「[グラントオペレーション](#)」を参照してください。[グラントの制約](#)を使用して、対称暗号化キーに対するグラントにある許可を制限することもできます。
- プリンシパルはアクセス許可を取得して、キーポリシーまたは IAM ポリシーからグラントを作成できます。ポリシーから `kms:CreateGrant` アクセス許可を取得したプリンシパルは、KMS キーの任意の[付与オペレーション](#)の許可を作成できます。これらのプリンシパルは、キーに対して付与している許可を持っている必要はありません。ポリシーで `kms:CreateGrant` アクセス許可を許可する場合は、[ポリシー条件](#)を使用してこの許可を制限します。
- プリンシパルは、グラントからグラントを作成する許可を取得することもできます。これらのプリンシパルは、何らかのポリシーからの他のアクセス許可を持っている場合でも、グラントを受けたアクセス許可のみを委任することができます。詳細については、「[CreateGrant アクセス許可の付与](#)」を参照してください。

グラントの概念

グラントを効果的に使用するには、AWS KMS が使用する用語と概念を理解する必要があります。

グラントの制約

グラントのアクセス許可を制限する条件。現在、は、[暗号化オペレーションのリクエストの暗号化コンテキスト](#)に基づく許可制約 AWS KMS をサポートしています。詳細については、「[グラントの制約の使用](#)」を参照してください。

グラント ID

KMS キーのグラントの一意の識別子。[キー識別子](#)と共にグラント ID を使用し、[RetireGrant](#) または [RevokeGrant](#) リクエストでグラントを識別できます。

グラントオペレーション

許可で許可できる AWS KMS オペレーション。他のオペレーションを指定した場合、`ValidationError` の例外により、[CreateGrant](#) リクエストは失敗します。これらは、[グラントトークン](#) を承認するオペレーションでもあります。これらのアクセス許可の詳細については、[AWS KMS アクセス許可](#) を参照してください。

これらのグラントオペレーションは、オペレーションを使用するアクセス許可を表します。したがって、`ReEncrypt` オペレーションの場合、`ReEncryptFrom`、`ReEncryptTo`、または両方の `ReEncrypt*` を指定できます。

グラントオペレーション:

- 暗号化オペレーション
 - [Decrypt](#)
 - [DeriveSharedSecret](#)
 - [暗号化](#)
 - [GenerateDataKey](#)
 - [GenerateDataKeyPair](#)
 - [GenerateDataKeyPairWithoutPlaintext](#)
 - [GenerateDataKeyWithoutPlaintext](#)
 - [GenerateMac](#)
 - [ReEncryptFrom](#)
 - [ReEncryptTo](#)
 - [Sign](#)
 - [検証](#)
 - [VerifyMac](#)
- その他のオペレーション
 - [CreateGrant](#)
 - [DescribeKey](#)
 - [GetPublicKey](#)
 - [RetireGrant](#)

許可するグラントオペレーションは、許可の KMS キーでサポートされている必要があります。サポートされていないオペレーションを指定すると、`ValidationError` の例外に

より、[CreateGrant](#) リクエストは失敗します。例えば、対称暗号化 KMS キーのグラントは、[Sign](#)、[Verify](#)、[GenerateMac](#) または [VerifyMac](#) オペレーションを許可できません。非対称 KMS キーのグラントは、データキーまたはデータキーペアを生成するいかなるオペレーションも許可できません。

グラントトークン

AWS KMS API は[結果整合性](#)モデルに従います。グラントを作成すると、変更が AWS KMS 全体に適用されるまでに若干の遅延が生じることがあります。通常、変更がシステム全体に反映されるまでに数秒もかかりませんが、場合によっては数分かかることがあります。グラントがシステム全体に完全に伝播される前に使用しようとする、アクセス拒否エラーが発生することがあります。グラントトークンを使用すると、グラントを参照し、グラントのアクセス許可をすぐに使用できます。

グラントトークンは、一意、非シークレット、可変長、base64 エンコードの、グラントを表す文字列です。グラントトークンを使用して、任意の[グラントオペレーション](#)でグラントを識別できます。ただし、トークン値はハッシュダイジェストであるため、グラントの詳細は明らかになりません。

グラントトークンは、グラントが AWS KMS 全体に完全に伝播されるまでにのみ使用されるように設計されています。その後、[被付与者プリンシパル](#)は、グラントトークンやその他のグラント限の証拠を提供することなく、グラントでアクセス許可を使用することができます。グラントトークンはいつでも使用できますが、グラントが最終的に一貫していると、はグラントトークンではなく、グラント AWS KMS を使用してアクセス許可を決定します。

例えば、次のコマンドでは、[GenerateDataKey](#) オペレーションを呼び出します。これは、グラントトークンを使用して、発信者 (被付与者プリンシパル) に、指定した KMS キーで `GenerateDataKey` を呼び出す許可を付与します。

```
$ aws kms generate-data-key \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --key-spec AES_256 \
  --grant-token $token
```

また、グラントトークンを使用して、グラントを管理するオペレーションでグラントを識別することもできます。例えば、[廃止プリンシパル](#)は、[RetireGrant](#) オペレーションの呼び出し時に、グラントトークンを使用できます。

```
$ aws kms retire-grant \
```

```
--grant-token $token
```

CreateGrant は、グラントトークンを返す唯一のオペレーションです。他のオペレーションまたは CreateGrant AWS KMS オペレーションの [CloudTrail ログイベント](#) からグラントトークンを取得することはできません。[ListGrants](#) および [ListRetirableGrants](#) オペレーションは、グラントトークンではなく [グラント ID](#) を返します。

詳細については、「[グラントトークンを使用する](#)」を参照してください。

被付与者プリンシパル

グラントで指定されたアクセス許可を取得する ID。各グラントはそれぞれ 1 つの被付与者プリンシパルを持っていますが、この被付与者プリンシパルは複数の ID を代表することができます。

被付与者プリンシパルは、AWS アカウント（ルート）、[IAM ユーザー](#)、[IAM ロール](#)、[フェデレーテッドロールまたはユーザー](#)、引き受けたロールユーザーなど、任意のプリンシパルにすることができます。被付与者プリンシパルは、KMS キーと同じアカウントか、別のアカウントにすることができます。ただし、被付与者プリンシパルを、[サービスプリンシパル](#)、[IAM グループ](#)、[AWS 組織](#)にすることはできません。

Note

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

(グラント) を廃止にする

グラントを終了します。アクセス許可の使用が終了したら、グラントを廃止にします。

グラントの取り消しと使用停止のどちらも、グラント限を削除します。ただし、使用停止はグラントで指定されたプリンシパルによって行われます。通常、取り消しはキー管理者が行います。詳細については、「[グラントの使用停止と取り消し](#)」を参照してください。

プリンシパルを使用停止にする

[グラントを廃止](#)にするプリンシパル。グラントで使用停止プリンシパルを指定できますが、必須ではありません。廃止プリンシパルは AWS アカウント、IAM ユーザー、IAM ロール、フェデレーテッドユーザー、引き受けたロールユーザーなど、任意のプリンシパルにすること

ができます。使用停止プリンシパルは、KMS キーと同じアカウントか、別のアカウントにすることができます。

Note

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

グラントで指定された廃止プリンシパルに加えて、グラントは、グラントが作成された AWS アカウント によって廃止できます。グラントで RetireGrant オペレーションが許可されている場合は、[被付与者プリンシパル](#)がグラントを廃止にできます。また、廃止 AWS アカウント するプリンシパルである AWS アカウント または、同じ の IAM プリンシパルに許可を廃止するアクセス許可を委任できます AWS アカウント。詳細については、「[グラントの使用停止と取り消し](#)」を参照してください。

(グラント) を取り消す

グラントを終了します。グラントを取り消して、グラントが許可するアクセス許可をアクティブに拒否します。

権限の取り消しと使用停止のどちらも、権限を削除します。ただし、使用停止はグラントで指定されたプリンシパルによって行われます。通常、取り消しはキー管理者が行います。詳細については、「[グラントの使用停止と取り消し](#)」を参照してください。

結果整合性 (グラント用)

AWS KMS API は[結果整合性](#)モデルに従います。グラントの作成、廃止、または取り消しを行うと、変更が AWS KMS 全体に適用されるまでに若干の遅延が生じることがあります。通常、変更がシステム全体に反映されるまでに数秒もかかりませんが、場合によっては数分かかることがあります。

想定外のエラーが発生する場合は、この短い遅延に注意する必要があります。例えば、新しい権限を管理したり、権限が全体で認識される前に新しい権限で権限を使用したりしようとすると AWS KMS、アクセス拒否エラーが発生する可能性があります。グラントを廃止にするか取り消す場合でも、被付与者プリンシパルは、グラントが完全に削除されるまで、そのアクセス許可を短い期間使用できる可能性があります。一般的な戦略はリクエストを再試行することであり、一部の AWS SDKs には自動バックオフと再試行ロジックが含まれています。

AWS KMS には、この短い遅延を軽減する機能があります。

- 新しいグラントでアクセス許可をすぐに使用するには、[グラントトークン](#)を使用します。グラントトークンを使用して、任意の[グラントのオペレーション](#)のグラントを参照できます。手順については、「[グラントトークンを使用する](#)」を参照してください。
- [CreateGrant](#) オペレーションには、再試行オペレーションが重複するグラントを作成することを防ぐ、Name パラメータがあります。

Note

サービスのすべてのエンドポイントが新しいグラント状態で更新されるまで、グラントトークンはグラントの有効性を優先します。ほとんどの場合、結果整合性は 5 分以内に取得されます。

詳細については、「[AWS KMS の結果整合性](#)」を参照してください。

AWS KMS グラントのベストプラクティス

AWS KMS では、許可を作成、使用、管理する際に以下のベストプラクティスを推奨しています。

- グラントのアクセス許可を、被付与者プリンシパルに必要なアクセス許可に制限します。[最小限の特権アクセス](#)の原則を使用します。
- IAM ロールなどの特定の被付与者プリンシパルを使用し、被付与者プリンシパルに、必要な API オペレーションのみを使用するアクセス許可を付与します。
- 暗号化コンテキストの[グラントの制約](#)を使用して、発信者が意図した目的のために KMS キーを使用していることを保証します。リクエストで暗号化コンテキストを使用してデータを保護する方法の詳細については、AWS セキュリティブログの「[AWS Key Management Service および EncryptionContext を使用して暗号化されたデータの整合性を保護する方法](#)」を参照してください。

Tip

可能な限り、[EncryptionContextEqual](#) のグラントの制約を使用します。[EncryptionContextSubset](#) のグラントの制約は、正しく使用するのが困難です。使用する必要がある場合は、ドキュメントをよく読み、グラントの制約をテストして、意図したとおりに動作することを確認してください。

- 重複するグラントを削除します。重複するグラントには、同じキー ARN、API アクション、被付与者プリンシパル、暗号化コンテキスト、名前などがあります。元のグラントを廃止にしたか取り消したにも関わらず、重複したグラントが残った場合、残った重複グラントは意図しない特権エスカレーションを構成します。CreateGrant リクエストの再試行時にグラントが重複しないようにするには、[Name パラメータ](#)を使用します。重複するグラントを検出するには、[ListGrants](#) オペレーションを使用します。誤って重複するグラントを作成した場合は、できるだけ早く廃止にするか、取り消します。

Note

[AWS マネージドキー](#)のグラントは重複しているようにみえますが、異なる被付与者プリンシパルを持っています。

通常、ListGrants レスポンスの GranteePrincipal フィールドには、グラントの被付与者プリンシパルが含まれます。ただし、グラントの被付与者プリンシパルが AWS サービスである場合、GranteePrincipal フィールドには[サービスプリンシパル](#)が含まれます。これは、複数の異なる被付与者プリンシパルを表す場合があります。

- グラントは、自動的に期限切れにならないことに注意してください。アクセス許可が不要になったら、すぐに[グラントの廃止または取り消し](#)をします。削除されないグラントは、暗号化されたリソースに対してセキュリティ上のリスクを引き起こす可能性があります。

グラントへのアクセスを制御する

キーポリシー、IAM ポリシー、グラントで、グラントを作成および管理するオペレーションへのアクセスを制御できます。グラントから CreateGrant 許可を取得したプリンシパルは、[より限定的な許可の付与](#)を行います。

API オペレーション	キーポリシーまたは IAM ポリシー	権限
CreateGrant	✓	✓
ListGrants	✓	-
ListRetirableGrants	✓	-

API オペレーション	キーポリシーまたは IAM ポリシー	権限
許可を使用停止にする	(制限あり。「 グラントの使用停止と取り消し 」を参照してください)	✓
RevokeGrant	✓	-

キーポリシーまたは IAM ポリシーを使用して、グラントを作成および管理するオペレーションへのアクセスを制御する場合、次のポリシー条件の 1 つ以上を使用してアクセス許可を制限できます。は、次のグラント関連の条件キーをすべて AWS KMS サポートします。詳細と例については、「[AWS KMS 条件キー](#)」を参照してください。

[kms:GrantConstraintType](#)

グラントに指定された[グラントの制約](#)が含まれている場合にのみ、プリンシパルにグラントの作成を許可します。

[kms:GrantIsForAWSResource](#)

[と統合された AWS サービスが AWS KMS](#)プリンシパルに代わってリクエストを送信するRevokeGrant場合にのみListGrants、プリンシパルが CreateGrant、または を呼び出すことを許可します。

[kms:GrantOperations](#)

プリンシパルにグラントの作成を許可しますが、グラントを指定されたオペレーションに制限します。

[kms:GranteePrincipal](#)

指定された[被付与者プリンシパル](#)に対してのみ、グラントの作成を許可します。

[kms:RetiringPrincipal](#)

グラントが特定の[使用停止プリンシパル](#)を指定する場合にのみ、プリンシパルにグラントの作成を許可します。

グラントの作成

グラントを作成する前に、グラントをカスタマイズするためのオプションを確認します。グラントの制約を使用して、グラントのアクセス許可を制限することができます。CreateGrant アクセス許可の付与も参照してください。グラントからグラントを作成する許可を取得したプリンシパルは、作成できるグラントが制限されます。

トピック

- [グラントの作成](#)
- [CreateGrant アクセス許可の付与](#)

グラントの作成

許可を作成するには、[CreateGrant](#) オペレーションを呼び出します。KMS キー、[被付与者プリンシパル](#)、許可された[グラントオペレーション](#)のリストを指定します。オプションの[使用停止プリンシパル](#)を指定することもできます。許可をカスタマイズするには、オプションの Constraints パラメータを使用して[許可の制約](#)を定義します。

グラントの作成、廃止、取り消しの際、変更が AWS KMS 全体で利用可能になるまで短い遅延が発生する場合があります (通常は 5 分未満)。詳細については、「[結果整合性 \(グラント用\)](#)」を参照してください。

例えば、次の CreateGrant コマンドは、keyUserRole ロールを引き受ける権限を持つユーザーに、指定された[対称 KMS キー](#)で [Decrypt](#) オペレーションを呼び出すことを許可するグラントを作成します。グラントでは、RetiringPrincipal パラメータを使用して、グラントを廃止できるプリンシパルを指定します。また、リクエスト内の[暗号化コンテキスト](#)に "Department": "IT" が含まれている場合にのみアクセス許可を付与する許可制約も含まれます。

```
$ aws kms create-grant \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \  
  --operations Decrypt \  
  --retiring-principal arn:aws:iam::111122223333:role/adminRole \  
  --constraints EncryptionContextSubset={Department=IT}
```

コードが CreateGrant オペレーションを再試行するか、[AWS リクエストを自動的に再試行する SDK](#) を使用する場合は、オプションの[名前](#)パラメータを使用して、重複グラントの作成を防止します。が既存のグラントと同じプロパティを持つグラントの CreateGrant リクエスト AWS KMS を取

得した場合、リクエストは再試行として認識され、新しいグラントは作成されません。Name 値を使用して、任意の AWS KMS オペレーションでグラントを識別することはできません。

 Important

グラント名には、機密情報や重要情報を含めないでください。これは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

```
$ aws kms create-grant \  
  --name IT-1234abcd-keyUserRole-decrypt \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \  
  --operations Decrypt \  
  --retiring-principal arn:aws:iam::111122223333:role/adminRole \  
  --constraints EncryptionContextSubset={Department=IT}
```

複数のプログラミング言語でのグラントの作成方法を示すコード例については、「[AWS SDK または CLI CreateGrant を使用する](#)」を参照してください。

グラントの制約の使用

[グラントの制約](#)は、グラントが被付与者プリンシパルに付与するアクセス許可の条件を設定します。グラントの制約は、[キーポリシー](#)または [IAM ポリシー](#)で、[条件キー](#)の代わりになります。各グラントの制約値には、最大 8 個の暗号化コンテキストペアを含めることができます。各グラントの制約の暗号化コンテキスト値は、384 文字を超えることはできません。

 Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

AWS KMS は、EncryptionContextEquals との EncryptionContextSubset 2 つの許可制約をサポートしています。どちらも、[暗号化オペレーションのリクエストで暗号化コンテキスト](#)の要件を確立します。

暗号化コンテキストのグラントの制約は、暗号化コンテキストパラメータを持つ[グラントオペレーション](#)で使用されるように設計されています。

- 暗号化コンテキストの制約は、対称暗号化 KMS キーのグラントのみで有効です。他の KMS キーを使用する暗号化オペレーションは、暗号化コンテキストをサポートしません。
- 暗号化コンテキストの制約は、DescribeKey および RetireGrant オペレーションでは無視されます。DescribeKey および RetireGrant には暗号化コンテキストパラメータはありませんが、暗号化コンテキストの制約を持つグラントにこれらのオペレーションを含めることができます。
- CreateGrant オペレーションのグラントで、暗号化コンテキストの制約を使用することができません。暗号化コンテキストの制約では、CreateGrant 許可で作成された任意のグラントが、同様に厳密またはより厳密な暗号化コンテキストの制約を持っている必要があります。

AWS KMS では、次の暗号化コンテキスト許可の制約がサポートされています。

EncryptionContextEquals

EncryptionContextEquals を使用して、許可されたリクエストの正確な暗号化コンテキストを指定します。

EncryptionContextEquals では、リクエストの暗号化コンテキストペアが、グラントの制約の暗号化コンテキストペアと、大文字と小文字の区別で完全に一致することを要求します。このペアは任意の順序で表示できますが、各ペアのキーと値を変更することはできません。

例えば、EncryptionContextEquals のグラントの制約が "Department": "IT" 暗号化コンテキストペアを要求する場合、グラントはリクエストの暗号化コンテキストが完全に "Department": "IT" である場合にのみ、指定されたタイプのリクエストを許可します。

EncryptionContextSubset

EncryptionContextSubset を使用して、リクエストに特定の暗号化コンテキストペアを含めるように要求します。

EncryptionContextSubset では、リクエストにグラントの制約 (完全な大文字と小文字を区別する一致) のすべての暗号化コンテキストペアを含むことを要求しますが、リクエストが追加の暗号化コンテキストペアを持っている可能性もあります。このペアは任意の順序で表示できますが、各ペアのキーと値を変更することはできません。

例えば、EncryptionContextSubset のグラントの制約が、Department=IT の暗号化コンテキストペアを要求する場合、グラントはリクエストの暗号化コンテキストが "Department": "IT" の場合、またはリクエストが "Department": "IT", "Purpose": "Test" のような他の暗号化コンテキストペアと共に "Department": "IT" を含む場合に、指定されたタイプのリクエストを許可します。

対称暗号化 KMS キーのグラントで暗号化コンテキストの制約を指定するには、[CreateGrant](#) オペレーションの `Constraints` パラメータを使用します。このコマンドが作成する許可では、`keyUserRole` ロールを引き受ける権限を持つユーザーに、[Decrypt](#) オペレーションを呼び出すためのアクセス許可を付与します。ただし、そのアクセス許可は `Decrypt` リクエストの暗号化コンテキストが、`"Department": "IT"` 暗号化コンテキストペアである場合にのみ有効です。

```
$ aws kms create-grant \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \  
  --operations Decrypt \  
  --retiring-principal arn:aws:iam::111122223333:role/adminRole \  
  --constraints EncryptionContextEquals={Department=IT}
```

したがって、グラントは以下のようになります。`keyUserRole` ロールに付与されるアクセス許可は、`Decrypt` リクエストがグラントの制約で指定された同じ暗号化コンテキストペアを使用する場合にのみ有効です。KMS キーでグラントを見つけるには、[ListGrants](#) オペレーションを使用します。

```
$ aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab  
{  
  "Grants": [  
    {  
      "Name": "",  
      "IssuingAccount": "arn:aws:iam::111122223333:root",  
      "GrantId":  
"abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",  
      "Operations": [  
        "Decrypt"  
      ],  
      "GranteePrincipal": "arn:aws:iam::111122223333:role/keyUserRole",  
      "Constraints": {  
        "EncryptionContextEquals": {  
          "Department": "IT"  
        }  
      },  
      "CreationDate": 1568565290.0,  
      "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
      "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole"  
    }  
  ]  
}
```

```
}
```

EncryptionContextEquals のグラントの制約を満たすために、Decrypt オペレーションのリクエストの暗号化コンテキストは、"Department": "IT" ペアである必要があります。被付与者プリンシパルからの次のようなリクエストは、EncryptionContextEquals のグラントの制約を満たします。

```
$ aws kms decrypt \  
  --key-id arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab\  
  --ciphertext-blob fileb://encrypted_msg \  
  --encryption-context Department=IT
```

グラントの制約が EncryptionContextSubset の場合、リクエストの暗号化コンテキストペアは、グラントの制約の暗号化コンテキストペアを含む必要がありますが、リクエストは他の暗号化コンテキストペアを含んでいる可能性もあります。次のグラントの制約は、リクエスト内の暗号化コンテキストペアの 1 つが "Department": "IT" であることを要求します。

```
"Constraints": {  
  "EncryptionContextSubset": {  
    "Department": "IT"  
  }  
}
```

被付与者プリンシパルからの次のリクエストは、この例の EncryptionContextEqual および EncryptionContextSubset のグラントの制約両方を満たします

```
$ aws kms decrypt \  
  --key-id arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \  
  --ciphertext-blob fileb://encrypted_msg \  
  --encryption-context Department=IT
```

ただし、被付与者プリンシパルからの次のようなリクエストは、EncryptionContextSubset のグラントの制約を満たしますが、EncryptionContextEquals のグラントの制約は満たしません。

```
$ aws kms decrypt \  
  --key-id arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \  
  --ciphertext-blob fileb://encrypted_msg
```

```
--ciphertext-blob fileb://encrypted_msg \  
--encryption-context Department=IT,Purpose=Test
```

AWS サービスは、多くの場合、で KMS キーを使用するアクセス許可を付与する許可で暗号化コンテキスト制約を使用します AWS アカウント。例えば、Amazon DynamoDB では、次のようなグラントを使用して、アカウントで DynamoDB の [AWS マネージドキー](#) を使用するアクセス許可を取得します。このグラント内の EncryptionContextSubset のグラントの制約により、要求内の暗号化コンテキストに "subscriberID": "111122223333" と "tableName": "Services" ペアが含まれる場合にのみ、グラントの許可が有効になります。このグラントの制約は、DynamoDB が AWS アカウントの特定のテーブルに対してのみ、指定された KMS キーを使用することをグラントが許可することを意味します。

この出力を取得するには、アカウントの AWS マネージドキー for DynamoDB で [ListGrants](#) オペレーションを実行します。

```
$ aws kms list-grants --key-id 0987dcba-09fe-87dc-65ba-ab0987654321  
  
{  
  "Grants": [  
    {  
      "Operations": [  
        "Decrypt",  
        "Encrypt",  
        "GenerateDataKey",  
        "ReEncryptFrom",  
        "ReEncryptTo",  
        "RetireGrant",  
        "DescribeKey"  
      ],  
      "IssuingAccount": "arn:aws:iam::111122223333:root",  
      "Constraints": {  
        "EncryptionContextSubset": {  
          "aws:dynamodb:tableName": "Services",  
          "aws:dynamodb:subscriberId": "111122223333"  
        }  
      },  
      "CreationDate": 1518567315.0,  
      "KeyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",  
      "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",  
      "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",  
      "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",
```

```
    "GrantId":  
      "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59"  
    }  
  ]  
}
```

CreateGrant アクセス許可の付与

グラントには、CreateGrant オペレーションを呼び出す許可を含めることができます。ただし、[被付与者プリンシパル](#)が CreateGrant を呼び出す許可をポリシーからではなくグラントから取得すると、その許可は制限されます。

- 被付与者プリンシパルは、親グラントで一部またはすべてのオペレーションを許可するグラントのみを作成できます。
- 作成されたグラントの[グラントの制約](#)は、少なくとも親グラントの制約と同じくらい厳密である必要があります。

これらの制限は、CreateGrant 許可をポリシーから取得するプリンシパルには適用されませんが、プリンシパルの許可は[ポリシー条件](#)によって制限されます。

例えば、被付与者プリンシパルが GenerateDataKey、Decrypt、および CreateGrant オペレーションを呼び出せるようにする許可について考えてみます。CreateGrant 許可、親グラントを許可するグラントを呼び出します。

```
# The original grant in a ListGrants response.  
{  
  "Grants": [  
    {  
      "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
      "CreationDate": 1572216195.0,  
      "GrantId":  
        "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",  
      "Operations": [  
        "GenerateDataKey",  
        "Decrypt",  
        "CreateGrant"  
      ],  
      "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole",  
      "Name": "",  
      "IssuingAccount": "arn:aws:iam::111122223333:root",  
    }  
  ]  
}
```

```

    "GranteePrincipal": "arn:aws:iam::111122223333:role/keyUserRole",
    "Constraints": {
      "EncryptionContextSubset": {
        "Department": "IT"
      }
    },
  ],
]
}

```

被付与者プリンシパル、exampleUser はこのアクセス許可を使用して、CreateGrant や Decrypt などの元のグラントで、指定されたオペレーションのサブセットを含むグラントを作成できます。子グラントに ScheduleKeyDeletion または ReEncrypt などの他のオペレーションを含めることはできません。

また、子グラントの[グラントの制約](#)は、親グラントの制約と同じか、より厳密である必要があります。例えば、子グラントは親グラントの EncryptionContextSubset 制約にペアを追加できますが、削除することはできません。子グラントは EncryptionContextSubset 制約を EncryptionContextEquals 制約に変更することはできますが、その逆はできません。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

例えば、被付与者プリンシパルは、親グラントから取得した CreateGrant 許可を使用して、次の子グラントを作成します。子グラントのオペレーションは、親グラントのオペレーションのサブセットであり、グラントの制約がより限定的です。

```

# The child grant in a ListGrants response.
{
  "Grants": [
    {
      "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "CreationDate": 1572249600.0,
      "GrantId": "fedcba9999c1e2e9876abcde6e9d6c9b6a1987650000abcee009abcdef40183f",
      "Operations": [
        "CreateGrant"
        "Decrypt"
      ]
    }
  ]
}

```



```
    "RetiringPrincipal": "arn:aws:iam::111122223333:user/exampleUser",
    "Name": "",
    "IssuingAccount": "arn:aws:iam::111122223333:root",
    "GranteePrincipal": "arn:aws:iam::111122223333:user/anotherUser",
    "Constraints": {
      "EncryptionContextEquals": {
        "Department": "IT"
      }
    },
  },
]
}
```

子グラントの被付与者プリンシパル、anotherUser は CreateGrant 許可を使用してグラントを作成できます。ただし、anotherUser が作成したグラントは親グラントまたはサブセット内のオペレーションを含める必要があり、グラントの制約は同じか、より厳密である必要があります。

グラントの表示

許可を表示するには、[ListGrants](#) オペレーションを使用します。グラントを適用する KMS キーを指定する必要があります。グラント ID または被付与者プリンシパルにより、グラントリストをフィルタリングすることもできます。その他の例については、「[AWS SDK または CLI ListGrantsで を使用する](#)」を参照してください。

特定の[廃止プリンシパル](#)を持つ AWS アカウント および リージョンのすべての許可を表示するには、[ListRetirableGrants](#) を使用します。レスポンスには、各グラントの詳細が含まれます。

Note

通常、ListGrants レスポンスの GranteePrincipal フィールドには、グラントの被付与者プリンシパルが含まれます。ただし、グラントの被付与者プリンシパルが AWS サービスである場合、GranteePrincipal フィールドには、複数の異なる被付与者[プリンシパル](#)を表すサービスプリンシパルが含まれます。

例えば、次のコマンドは KMS キーのすべてのグラントを一覧表示します。

```
$ aws kms list-grants --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "Grants": [
```

```
{
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "CreationDate": 1572216195.0,
  "GrantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a",
  "Constraints": {
    "EncryptionContextSubset": {
      "Department": "IT"
    }
  },
  "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole",
  "Name": "",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "GranteePrincipal": "arn:aws:iam::111122223333:user/exampleUser",
  "Operations": [
    "Decrypt"
  ]
}
```

グラントトークンを使用する

AWS KMS API は[結果整合性](#)モデルに従います。グラントの作成時、グラントがすぐに有効にならないことがあります。変更が AWS KMS 全体に適用されるまでに若干の遅延が生じることがあります。通常、変更がシステム全体に反映されるまでに数秒もかかりませんが、場合によっては数分かかることがあります。システム全体で変更が完全に伝播されると、被付与者プリンシパルはグラントトークンやグラントの証拠を指定せずに、グラントのアクセス許可を使用できます。ただし、まだすべてのに知られていないほど新しい権限がある場合 AWS KMS、リクエストは `AccessDeniedException` エラーで失敗する可能性があります。

新しいグラントでアクセス許可をすぐに使用するには、グラントの[グラントトークン](#)を使用します。[CreateGrant](#) オペレーションによって返されるグラントトークンを保存します。次に、AWS KMS オペレーションのリクエストでグラントトークンを送信します。グラントトークンは任意の AWS KMS [グラントオペレーション](#)に送信でき、同じリクエストで複数のグラントトークンを送信できます。

次の例では、`CreateGrant` オペレーションを使用して、[GenerateDataKey](#) および [Decrypt](#) オペレーションを許可するグラントを作成します。これは、`CreateGrant` が `token` 変数で返すグラント

トークンを保存します。次に、GenerateDataKey オペレーションへの呼び出しで、token 変数の GrantToken を使用します。

```
# Create a grant; save the grant token
$ token=$(aws kms create-grant \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --grantee-principal arn:aws:iam::111122223333:user/appUser \
  --retiring-principal arn:aws:iam::111122223333:user/acctAdmin \
  --operations GenerateDataKey Decrypt \
  --query GrantToken \
  --output text)

# Use the grant token in a request
$ aws kms generate-data-key \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --key-spec AES_256 \
  --grant-tokens $token
```

アクセス許可を持つプリンシパルは、その GrantToken が AWS KMS 全体で利用可能になる前であっても、GrantToken を使用して新しい GrantToken を廃止することができます。(RevokeGrant オペレーションは GrantToken を承認しません)。詳細については、「[GrantToken の使用停止と取り消し](#)」を参照してください。

```
# Retire the grant
$ aws kms retire-grant --grant-token $token
```

GrantToken の使用停止と取り消し

GrantToken を削除するには、GrantToken の廃止または取り消しをします。

[RetireGrant](#) および [RevokeGrant](#) オペレーションは、互いに非常によく似ています。どちらのオペレーションも GrantToken を削除します。これにより、GrantToken が許可しているアクセス許可が削除されます。これらのオペレーションの主な違いは、オペレーションの認可方法です。

RevokeGrant

ほとんどの AWS KMS オペレーションと同様に、RevokeGrant オペレーションへのアクセスは [キーポリシー](#) と [IAM ポリシー](#) によって制御されます。[RevokeGrant](#) API は、任意のプリンシパルから kms:RevokeGrant アクセス許可で呼び出すことができます。このアクセス許可は、キー管理者に付与される標準のアクセス許可に含まれています。通常、管理者は GrantToken を取り消して、GrantToken が許可するアクセス許可を拒否します。

RetireGrant

グラントでは、グラントを廃止にできる管理者を決定できます。この設計により、キーポリシーや IAM ポリシーを変更することなく、グラントのライフサイクルを制御できます。通常、許可の使用を終了したら、グラントを廃止にします。

グラントは、グラントで指定されたオプションの[廃止プリンシパル](#)により廃止にできます。[被付与者プリンシパル](#)もグラントを廃止にできますが、RetireGrant オペレーションを含むプリンシパルまたはグラントも同時に廃止にする場合に限られます。バックアップとして、グラントが作成された AWS アカウント はグラントを廃止できます。

IAM ポリシーで利用できる kms:RetireGrant アクセス許可がありますが、ユーティリティは限られています。グラントで指定されたプリンシパルは、kms:RetireGrant アクセス許可なしでグラントを廃止にできます。kms:RetireGrant アクセス許可だけでは、プリンシパルにグラントの廃止を許可できません。アクセスkms:RetireGrant許可は、[キーポリシー](#)または[リソースコントロールポリシー](#)では有効ではありません。

- 許可を廃止するアクセス許可を拒否するには、IAM ポリシーの アクセスkms:RetireGrant許可を持つDenyアクションを使用できます。
- KMS キーを所有 AWS アカウント する は、アカウントの IAM プリンシパルにアクセスkms:RetireGrant許可を委任できます。
- 廃止プリンシパルが異なる場合 AWS アカウント、他のアカウントの管理者は kms:RetireGrant を使用して、そのアカウントの IAM プリンシパルに付与を廃止するアクセス許可を委任できます。

AWS KMS API は[結果整合性](#)モデルに従います。グラントの作成、廃止、または取り消しを行うと、変更が AWS KMS全体に適用されるまでに若干の遅延が生じることがあります。通常、変更がシステム全体に反映されるまでに数秒もかかりませんが、場合によっては数分かかることがあります。新しい権限をすぐに削除する必要がある場合は、その権限トークン AWS KMS[を使用して権限](#)を廃止します。グラントトークンを使用してグラントを取り消すことはできません。

の条件キー AWS KMS

AWS KMS リソースへのアクセスを制御する[キーポリシー](#)と [IAM ポリシー](#)で条件を指定できます。ポリシーステートメントは、条件が true の場合にのみ有効です。例えば、特定の日付の後にのみ適用されるポリシーステートメントが必要になる場合があります。また、特定の値が API リクエストに表示されているときにのみ、ポリシーステートメントによるアクセスコントロールが必要になる場合もあります。

条件を指定するには、[IAM 条件演算子](#)を指定して、ポリシーステートメントの [Condition 要素](#)で条件キーを使用します。一部の条件キーは一般的に適用され、その他の条件キーは固有です AWS KMS。

条件キー値は、AWS KMS キーポリシーと IAM ポリシーの文字とエンコーディングルールに従う必要があります。キーポリシードキュメントのルールに関する詳細については、「[キーポリシー形式](#)」を参照してください。IAM ポリシードキュメントのルールに関する詳細については、「IAM ユーザーガイド」の「[IAM 名前の要件](#)」を参照してください。

トピック

- [AWS グローバル条件キー](#)
- [AWS KMS 条件キー](#)
- [AWS KMSAWS Nitro Enclaves の条件キー](#)

AWS グローバル条件キー

AWS は、[アクセス](#)コントロールに IAM を使用するすべての AWS サービスのポリシー条件キーのセットであるグローバル条件キーを定義します。は、すべてのグローバル条件キー AWS KMS をサポートします。AWS KMS キーポリシーと IAM ポリシーで使用できます。

例えば、[aws:PrincipalArn](#) グローバル条件キーを使用すると、リクエスト内のプリンシパルが条件キーバリューの Amazon リソースネーム (ARN) で表されている場合にのみ、AWS KMS key (KMS キー) へのアクセスを許可できます。で[属性ベースのアクセスコントロール](#) (ABAC) をサポートするには AWS KMS、IAM ポリシーで [aws:ResourceTag/tag-key](#) グローバル条件キーを使用して、特定のタグを持つ KMS キーへのアクセスを許可できます。

プリンシパルがサービスプリンシパルであるポリシーで、AWS サービスが混乱した代理として使用されないようにするには、[aws:SourceArn](#)または [aws:SourceAccount](#) グローバル条件キーを使用できます。詳細については、「[aws:SourceArn または aws:SourceAccount 条件キーの使用](#)」を参照してください。

利用可能なリクエストのタイプを含む AWS グローバル条件キーの詳細については、IAM ユーザーガイドの[AWS 「グローバル条件コンテキストキー」](#)を参照してください。IAM ポリシーでグローバル条件キーを使用する例については、IAM ユーザーガイドの[リクエストへのアクセスの制御](#)および[タグキーの制御](#)を参照してください。

以下のトピックでは、IP アドレスと VPC エンドポイントに基づく条件キーを使用するための特別なガイダンスを提供します。

トピック

- [AWS KMS アクセス許可を持つポリシーでの IP アドレス条件の使用](#)
- [AWS KMS アクセス許可を持つポリシーでの VPC エンドポイント条件の使用](#)
- [IAM および AWS KMS キーポリシーでの IPv6 アドレスの使用](#)

AWS KMS アクセス許可を持つポリシーでの IP アドレス条件の使用

を使用して AWS KMS、[統合 AWS サービス](#)内のデータを保護できます。ただし、アクセスを許可または拒否するのと同じポリシーステートメントで [IP アドレス条件演算子](#)または `aws:SourceIp` 条件キーを指定する場合は注意が必要です AWS KMS。たとえば、のポリシー [AWS: ソース IP AWS に基づいて へのアクセスを拒否すると](#)、AWS アクションは指定された IP 範囲からのリクエストに制限されます。

次のシナリオを考えてみます。

1. [AWS 「: ソース IP AWS に基づいて へのアクセスを拒否する」](#) に示されているようなポリシーを IAM アイデンティティにアタッチします。 `aws:SourceIp` 条件キーの値は、ユーザーの会社の IP アドレス範囲に設定します。この IAM アイデンティティには、Amazon EBS、Amazon EC2、AWS KMSの使用を許可する他のポリシーがアタッチされています。
2. アイデンティティは、暗号化された EBS ボリュームを EC2 インスタンスにアタッチしようとしています。このアクションは、関連するすべてのサービスを使用するためのアクセス権限がユーザーに付与されているにもかかわらず、承認エラーが発生して失敗します。

ボリュームの暗号化されたデータキーを復号 AWS KMS する へのリクエストは、Amazon EC2 インフラストラクチャに関連付けられている IP アドレスから送信されるため、ステップ 2 は失敗します。成功させるには、リクエストは、元のユーザーの IP アドレスからリクエストが送られてこなければなりません。ステップ 1 のポリシーでは、指定されたもの以外の IP アドレスからのすべてのリクエストが明示的に拒否されるため、Amazon EC2 は EBS ボリュームの暗号化されたデータキーを復号化する権限を拒否されます。

また、リクエストが [Amazon VPC エンドポイント](#) から送信される場合、`aws:SourceIP` 条件キーは無効です。リクエストを [AWS KMS VPC エンドポイント](#) などの VPC エンドポイントに制限するには、`aws:SourceVpce` または `aws:SourceVpc` 条件キーを使用します。詳細については、[Amazon VPC ユーザーガイド](#) の「VPC エンドポイント - エンドポイントの使用の管理」を参照してください。

AWS KMS アクセス許可を持つポリシーでの VPC エンドポイント条件の使用

は、[AWS KMS PrivateLink を搭載した Amazon Virtual Private Cloud \(Amazon VPC\) エンドポイントをサポートしています](#)。[AWS PrivateLink](#) キーポリシーと IAM ポリシーで次の[グローバル条件キー](#)を使用して、リクエストが VPC から送信されたとき、または VPC エンドポイントを使用する場合の AWS KMS リソースへのアクセスを制御できます。詳細については、「[VPC エンドポイントを使用して AWS KMS リソースへのアクセスを制御する](#)」を参照してください。

- `aws:SourceVpc` は、指定した VPC からのリクエストにアクセスを制限します。
- `aws:SourceVpce` は、指定した VPC エンドポイントからのリクエストにアクセスを制限します。

これらの条件キーを使用して KMS キーへのアクセスを制御すると、が AWS KMS ユーザーに代わって使用する AWS サービスへのアクセスが誤って拒否される可能性があります。

[IP アドレス条件キー](#)の例のような状況にならないように注意してください。KMS キーのリクエストを VPC または VPC エンドポイントに制限すると、Amazon S3 や Amazon EBS などの統合サービス AWS KMS からの の呼び出しが失敗する可能性があります。ソースリクエストの最初の送信元が VPC 内または VPC エンドポイントであっても、これは発生することがあります。

IAM および AWS KMS キーポリシーでの IPv6 アドレスの使用

IPv6 AWS KMS 経由で にアクセスする前に、IP アドレス制限を含むキーポリシーと IAM ポリシーが更新され、IPv6 アドレス範囲が含まれていることを確認してください。IPv6 アドレスを処理するように更新されていない IP ベースのポリシーでは、クライアントが IPv6 の使用を開始すると、クライアントが誤ってアクセスを失ったり、アクセスを取得したりする可能性があります。KMS アクセスコントロールの一般的なガイダンスについては、「」を参照してください[KMS キーのアクセスとアクセス許可](#)。KMS とデュアルスタックのサポートについては、「」を参照してください[デュアルスタックエンドポイントのサポート](#)。

Important

これらのステートメントでは、アクションは許可されません。これらのステートメントを、特定のアクションを許可する他のステートメントと組み合わせて使用します。

次のステートメントは、IPv4 アドレス `192.0.2.*` の範囲から送信されるリクエストに対するすべての KMS アクセス許可へのアクセスを明示的に拒否します。この範囲外の IP アドレスは、KMS アク

セス許可を明示的に拒否されません。すべての IPv6 アドレスが拒否範囲外であるため、このステートメントは IPv6 アドレスの KMS アクセス許可を明示的に拒否しません。

```
{
  "Sid": "DenyKMSPermissions",
  "Effect": "Deny",
  "Action": [
    "kms:*"
  ],
  "Resource": "*",
  "Condition": {
    "NotIpAddress": {
      "aws:SourceIp": [
        "192.0.2.0/24"
      ]
    }
  }
}
```

次の例に示すように、Condition要素を変更してIPv4 (192.0.2.0/24) アドレス範囲と IPv6 (2001:db8:1234::/32) アドレス範囲の両方を拒否できます。

```
{
  "Sid": "DenyKMSPermissions",
  "Effect": "Deny",
  "Action": [
    "kms:*"
  ],
  "Resource": "*",
  "Condition": {
    "NotIpAddress": {
      "aws:SourceIp": [
        "192.0.2.0/24",
        "2001:db8:1234::/32"
      ]
    }
  }
}
```

AWS KMS 条件キー

AWS KMS には、キーポリシーと IAM ポリシーで使用できる一連の条件キーが用意されています。これらの条件キーはに固有です AWS KMS。例えば、`kms:EncryptionContext:context-key` 条件キーは、対称暗号化 KMS キーへのアクセスを制御するときに特定の[暗号化コンテキスト](#)を要求するために使用できます。

API オペレーションリクエストの条件

多くの AWS KMS 条件キーは、AWS KMS オペレーションのリクエストのパラメータの値に基づいて KMS キーへのアクセスを制御します。例えば、IAM ポリシーの [kms:KeySpec](#) 条件キーを使用すると、CreateKey リクエストの KeySpec パラメータ値が RSA_4096 の場合にのみ、[CreateKey](#) オペレーションの使用を許可できます。

このタイプの条件は、パラメータのデフォルト値を使用する場合など、リクエストにパラメータが表示されない場合でも機能します。例えば、[kms:KeySpec](#) 条件キーを使用すると、KeySpec パラメータ値が SYMMETRIC_DEFAULT (デフォルト値) の場合にのみ、CreateKey オペレーションの使用をユーザーに許可できます。この条件では、SYMMETRIC_DEFAULT 値を持つ KeySpec パラメータを持つリクエストと、KeySpec パラメータを持たないリクエストが許可されます。

API オペレーションで使用される KMS キーの条件

一部の AWS KMS 条件キーは、オペレーションで使用される KMS キーのプロパティに基づいて、オペレーションへのアクセスを制御できます。例えば、[kms:KeyOrigin](#) 条件を使用すると、KMS キーの Origin が AWS_KMS の場合にのみ、プリンシパルが [GenerateDataKey](#) を KMS キーで呼び出すことができます。この方法で条件キーを使用できるかどうかを確認するには、条件キーの説明を参照してください。

このオペレーションは KMS キーリソースオペレーションである必要があります。つまり、特定の KMS キーに認可されるオペレーションです。KMS キーリソースオペレーションを識別するには、[アクションとリソースの表](#)で、オペレーションの Resources 列の KMS key の値を探します。[ListKeys](#) など、特定の KMS キーリソースに対して認可されていないオペレーションでこのタイプの条件キーを使用すると、条件が満たされないため、アクセス許可は有効になりません。ListKeys オペレーションの認可に関与する KMS キーリソースおよび KeySpec プロパティはありません。

以下のトピックでは、各 AWS KMS 条件キーについて説明し、ポリシー構文を示すポリシーステートメントの例を示します。

条件キーで集合演算子を使用する

ポリシー条件が、リクエスト内のタグのセットとポリシー内のタグのセットなど、2つの値のセットを比較する場合、セットを比較する AWS 方法を に指示する必要があります。この目的のために、IAM は、2つの集合演算子、ForAnyValue および ForAllValues を定義します。集合演算子は、それらを必要とする複数值を持つ条件キーでのみ使用します。単一値の条件キーで集合演算子を使用しないでください。ポリシーステートメントは必ず、本稼働環境での使用前に完全にテストしてください。

条件キーは単一値または複数值です。AWS KMS 条件キーが単一値か複数值かを判断するには、条件キーの説明の「値タイプ」列を参照してください。

- 単一値の条件キーは認可コンテキスト (リクエストまたはリソース) に、最大で 1 つの値を持ちます。たとえば、各 API コールは 1 つのみから発信できるため AWS アカウント、[kms:CallerAccount](#) は単一の値条件キーです。単一値の条件キーで集合演算子を使用しないでください。
- 複数值の条件キーでは、認可コンテキスト (リクエストまたはリソース) に複数の値があります。例えば、各 KMS キーは複数のエイリアスを持つことができるため、[kms:ResourceAliases](#) は複数值を持つことができます。複数值の条件キーには集合演算子が必要です。

単一値と複数值の条件キーの違いは、ポリシー条件の値の数ではなく、認可コンテキストの値の数であることに注意してください。

Warning

単一値の条件キーで集合演算子を使用すると、過度に許可される (または過度に制限される) ポリシーステートメントが作成される可能性があります。集合演算子は、複数值の条件キーでのみ使用してください。

kms:EncryptionContext:context-key または aws:RequestTag/tag-key 条件キーを持つ ForAllValues 集合演算子を含むポリシーを作成または更新すると、は次のエラーメッセージ AWS KMS を返します。

```
OverlyPermissiveCondition: Using the ForAllValues set operator with a single-valued condition key matches requests without the specified [encryption context or tag] or with an unspecified [encryption context or tag]. To fix, remove ForAllValues.
```

ForAnyValue および ForAllValues 集合演算子の詳細については、IAM ユーザーガイドの[複数のキーと値の使用](#)を参照してください。ForAllValues 集合演算子を単一値の条件で使用するリスク

の詳細については、IAM ユーザーガイドの [Security Warning – ForAllValues with single valued key](#) を参照してください。

トピック

- [kms:BypassPolicyLockoutSafetyCheck](#)
- [kms:CallerAccount](#)
- [kms:CustomerMasterKeySpec](#) (非推奨)
- [kms:CustomerMasterKeyUsage](#) (非推奨)
- [kms:DataKeyPairSpec](#)
- [kms:EncryptionAlgorithm](#)
- [kms:EncryptionContext: context-key](#)
- [kms:EncryptionContextKeys](#)
- [kms:ExpirationModel](#)
- [kms:GrantConstraintType](#)
- [kms:GrantIsForAWSResource](#)
- [kms:GrantOperations](#)
- [kms:GranteePrincipal](#)
- [kms:KeyAgreementAlgorithm](#)
- [kms:KeyOrigin](#)
- [kms:KeySpec](#)
- [kms:KeyUsage](#)
- [kms:MacAlgorithm](#)
- [kms:MessageType](#)
- [kms:MultiRegion](#)
- [kms:MultiRegionKeyType](#)
- [kms:PrimaryRegion](#)
- [kms:ReEncryptOnSameKey](#)
- [kms:RequestAlias](#)
- [kms:ResourceAliases](#)

- [kms:ReplicaRegion](#)
- [kms:RetiringPrincipal](#)
- [kms:RotationPeriodInDays](#)
- [kms:ScheduleKeyDeletionPendingWindowInDays](#)
- [kms:SigningAlgorithm](#)
- [kms:ValidTo](#)
- [kms:ViaService](#)
- [kms:WrappingAlgorithm](#)
- [kms:WrappingKeySpec](#)

kms:BypassPolicyLockoutSafetyCheck

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:BypassPolicyLockoutSafetyCheck	ブール値	単一値	CreateKey PutKeyPolicy	IAM ポリシーのみ キーポリシーと IAM ポリシー

kms:BypassPolicyLockoutSafetyCheck 条件キーは、リクエストの [パラメータの値に基づき](#)、CreateKey [オペレーションおよび](#) PutKeyPolicyBypassPolicyLockoutSafetyCheck オペレーションへのアクセスを制御します。

次の IAM ポリシーステートメントの例では、CreateKey リクエストの BypassPolicyLockoutSafetyCheck パラメータ値が true. の場合に、KMS キーを作成するアクセス許可を拒否することで、ユーザーがポリシーのロックアウト安全チェックを回避できないようにします。

```
{
  "Effect": "Deny",
  "Action": [
    "kms:CreateKey",
```

```
    "kms:PutKeyPolicy"
  ],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:BypassPolicyLockoutSafetyCheck": true
    }
  }
}
```

IAM ポリシーまたはキーポリシーで `kms:BypassPolicyLockoutSafetyCheck` 条件キーを使用して、`PutKeyPolicy` オペレーションへのアクセスを制御することもできます。次のキーポリシーのポリシーステートメントの例では、KMS キーのポリシーを変更する際に、ユーザーがポリシーのロックアウト安全チェックを回避できないようにします。

このポリシーステートメントでは、明示的に `Deny` を使用せずに、[Null 条件演算子](#)とともに `Allow` を使用し、リクエストに `BypassPolicyLockoutSafetyCheck` パラメータが含まれていない場合にのみアクセスを許可します。パラメータが使用されていない場合、デフォルト値は `false` です。この弱いポリシーステートメントは、バイパスが必要な限定された状況では上書きされる場合があります。

```
{
  "Effect": "Allow",
  "Action": "kms:PutKeyPolicy",
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:BypassPolicyLockoutSafetyCheck": true
    }
  }
}
```

以下の資料も参照してください。

- [kms:KeySpec](#)
- [kms:KeyOrigin](#)
- [kms:KeyUsage](#)

kms:CallerAccount

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:CallerAccount	String	単一値	KMS キーリソースのオペレーション カスタムキーストアのオペレーション	キーポリシーと IAM ポリシー

この条件キーを使用して、AWS アカウントのすべてのアイデンティティ (ユーザーおよびロール) へのアクセスを許可または拒否できます。キーポリシーでは、Principal 要素を使って、ポリシーステートメントが適用される ID を指定できます。Principal 要素の構文では、AWS アカウントのすべてのアイデンティティを指定することはできません。ただし、この条件キーをすべての ID AWS を指定する Principal 要素と組み合わせることで、この効果を実現できます。

これを使用して、任意の KMS キーリソースオペレーション、つまり特定の KMS キーを使用する AWS KMS オペレーションへのアクセスを制御できます。KMS キーリソースオペレーションを識別するには、[アクションとリソースの表](#)で、オペレーションの Resources 列の KMS key の値を探します。また、[カスタムキーストア](#)を管理するオペレーションにも有効です。

例えば、次のポリシーステートメントは、kms:CallerAccount 条件キーを使用する方法を示します。このポリシーステートメントは、Amazon EBS の AWS マネージドキー のキーポリシーにあります。すべての ID AWS を指定する Principal 要素と kms:CallerAccount 条件キーを組み合わせ、AWS アカウント 111122223333 のすべての ID へのアクセスを効果的に許可します。Amazon EBS 経由のリクエストのみを許可することで、アクセス許可をさらに制限するための追加の AWS KMS 条件キー (kms:ViaService) が含まれています。詳細については、「[kms:ViaService](#)」を参照してください。

```
{
  "Sid": "Allow access through EBS for all principals in the account that are
authorized to use EBS",
  "Effect": "Allow",
  "Principal": {"AWS": "*"},
  "Condition": {
```



```
"StringEquals": {
  "kms:CallerAccount": "111122223333",
  "kms:ViaService": "ec2.us-west-2.amazonaws.com"
},
"Action": [
  "kms:Encrypt",
  "kms:Decrypt",
  "kms:ReEncrypt*",
  "kms:GenerateDataKey*",
  "kms:CreateGrant",
  "kms:DescribeKey"
],
"Resource": "*"
}
```

kms:CustomerMasterKeySpec (非推奨)

kms:CustomerMasterKeySpec 条件キーは非推奨です。代わりに、[kms:KeySpec](#) 条件キーを使用します。

kms:CustomerMasterKeySpec および kms:KeySpec 条件キーは同じように機能します。名前だけが異なります。kms:KeySpec を使用することをお勧めします。ただし、変更が中断されないように、は両方の条件キー AWS KMS をサポートします。

kms:CustomerMasterKeyUsage (非推奨)

kms:CustomerMasterKeyUsage 条件キーは非推奨です。代わりに、[kms:KeyUsage](#) 条件キーを使用します。

kms:CustomerMasterKeyUsage および kms:KeyUsage 条件キーは同じように機能します。名前だけが異なります。kms:KeyUsage を使用することをお勧めします。ただし、変更が中断されないように、は両方の条件キー AWS KMS をサポートします。

kms:DataKeyPairSpec

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:DataKeyPairSpec	String	単一値	GenerateDataKeyPair	キーポリシーと IAM ポリシー

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
			GeneratedDataKeyPairWithoutPlaintext	

この条件キーを使用して、リクエストの `KeyPairSpec` パラメータ値に基づく、[GenerateDataKeyPair](#) および [GenerateDataKeyPairWithoutPlaintext](#) オペレーションへのアクセスを制御します。例えば、特定のタイプのデータキーペアのみを生成することをユーザーに許可できます。

次のキーポリシーステートメントの例では、`kms:DataKeyPairSpec` 条件キーを使用して、ユーザーが KMS キーを使用して RSA データキーペアのみを生成できるようにします。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:GenerateDataKeyPair",
    "kms:GenerateDataKeyPairWithoutPlaintext"
  ],
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:DataKeyPairSpec": "RSA*"
    }
  }
}
```

以下の資料も参照してください。

- [kms:KeySpec](#)
- [the section called “kms:EncryptionAlgorithm”](#)
- [the section called “kms:EncryptionContext: context-key”](#)
- [the section called “kms:EncryptionContextKeys”](#)

kms:EncryptionAlgorithm

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Encry ptionAlgo rithm	String	単一値	Decrypt Encrypt Generated ataKey Generated ataKeyPair Generated ataKeyPai rWithoutP laintext Generated ataKeyWit houtPlain text ReEncrypt	キーポリシーと IAM ポリシー

kms:EncryptionAlgorithm 条件キーを使用して、オペレーションで使用される暗号化アルゴリズムに基づいて暗号化オペレーションへのアクセスを制御できます。[Encrypt](#)、[Decrypt](#)、および [ReEncrypt](#) オペレーションの場合、リクエストの [EncryptionAlgorithm](#) パラメータ値に基づいてアクセスを制御します。データキーとデータキーペアを生成するオペレーションでは、データキーの暗号化に使用される暗号化アルゴリズムに基づいてアクセスを制御します。

この条件キーは、非対称 KMS キーペアのパブリックキーによる暗号化など AWS KMS、 の外部で実行されるオペレーションには影響しません AWS KMS。

リクエスト内の暗号化アルゴリズムパラメータ

ユーザーが KMS キーで特定の暗号化アルゴリズムのみを使用できるようにするには、Deny 効果と StringNotEquals 条件演算子を含むポリシーステートメントを使用します。例えば、以下のキーポリシーステートメント例は、リクエスト内の暗号化アルゴリズムが RSAES_OAEP_SHA_256 (RSA KMS キーで使用される非対称暗号化アルゴリズム) 以外の場合に、ExampleRole ロールを引き受けることができるプリンシパルが、指定された暗号化オペレーションでこの KMS キーを使用することを禁止します。

ユーザーが特定の暗号化アルゴリズムを使用できるようにするポリシーステートメントとは異なり、このような二重否定を持つポリシーステートメントは、この KMS キーに対する他のポリシーおよび権限によって、このロールが他の暗号化アルゴリズムを使用することを防止します。このポリシーステートメントの Deny は、Allow 効果を持つキーポリシーまたは IAM ポリシーよりも優先され、この KMS キーおよびそのプリンシパルのすべての権限よりも優先されます。

```
{
  "Sid": "Allow only one encryption algorithm with this asymmetric KMS key",
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*"
  ],
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "kms:EncryptionAlgorithm": "RSAES_OAEP_SHA_256"
    }
  }
}
```

オペレーションに使用される暗号化アルゴリズム

kms:EncryptionAlgorithm 条件キーを使用して、リクエストでアルゴリズムが指定されていない場合でも、オペレーションで使用される暗号化アルゴリズムに基づいてオペレーションへのアクセスを制御することもできます。これにより、デフォルト値のためリクエストで指定されない可能性のある SYMMETRIC_DEFAULT アルゴリズムを要求または禁止することができます。

この機能により、`kms:EncryptionAlgorithm` 条件キーを使用して、データキーとデータキーペアを生成するオペレーションへのアクセスを制御することができます。これらのオペレーションは、対称暗号化 KMS キーと `SYMMETRIC_DEFAULT` アルゴリズムのみを使用します。

例えば、この IAM ポリシーは、プリンシパルを対称暗号化に制限します。リクエストで指定された、またはオペレーションで使用される暗号化アルゴリズムが `SYMMETRIC_DEFAULT` でない限り、暗号化オペレーションのサンプルアカウントにある KMS キーへのアクセスを拒否します。GenerateDataKey* が [GenerateDataKey](#)、[GenerateDataKeyWithoutPlaintext](#)、[GenerateDataKeyPair](#)、[GenerateDataKeyPairWithoutPlaintext](#) をアクセス許可へ追加することを含みます。この条件は、常に対称暗号化アルゴリズムを使用するため、これらのオペレーションには影響しません。

```
{
  "Sid": "AllowOnlySymmetricAlgorithm",
  "Effect": "Deny",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringNotEquals": {
      "kms:EncryptionAlgorithm": "SYMMETRIC_DEFAULT"
    }
  }
}
```

以下の資料も参照してください。

- [the section called “kms:MacAlgorithm”](#)
- [kms:SigningAlgorithm](#)

kms:EncryptionContext: context-key

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:EncryptionContext: <i>context-key</i>	String	単一値	CreateGrant Encrypt Decrypt GeneratedataKey GeneratedataKeyPair GeneratedataKeyPairWithoutPlaintext GeneratedataKeyWithoutPlaintext ReEncrypt RetireGrant	キーポリシーとIAM ポリシー

kms:EncryptionContext:context-key 条件キーを使用して、[対称暗号化 KMS キー](#)へのアクセスを、[暗号化オペレーション](#)に対するリクエスト内の[暗号化コンテキスト](#)に基づいて制御することができます。暗号化コンテキストペアのキーと値の両方を評価するには、この条件キーを使用します。暗号化コンテキストキーのみを評価するか、キーまたは値に関係なく暗号化コンテキストを要求するには、[kms:EncryptionContextKeys](#) 条件キーを使用します。

Note

条件キーの値は、キーポリシーと IAM ポリシーに関する文字ルールに従う必要があります。暗号化コンテキストで有効な文字の中には、ポリシーでは有効にならないものもあります。この条件キーは、すべての有効な暗号化コンテキスト値を表現するために使用できない場合があります。キーポリシードキュメントのルールに関する詳細については、「[キーポリシー形式](#)」を参照してください。IAM ポリシードキュメントのルールに関する詳細については、「IAM ユーザーガイド」の「[IAM 名前の要件](#)」を参照してください。

[非対称 KMS キー](#)または [HMAC KMS キー](#)を使用する暗号化オペレーションで暗号化コンテキストを指定することはできません。非対称アルゴリズムと MAC アルゴリズムは、暗号化コンテキストをサポートしません。

kms:EncryptionContext:context-key 条件キーを使用するには、*context-key* プレースホルダーを暗号化コンテキストキーと置き換えます。*#####* プレースホルダーを暗号化コンテキスト値と置き換えます。

```
"kms:EncryptionContext:context-key": "context-value"
```

例えば、次の条件キーでは、キーが AppName、値が ExampleApp (AppName = ExampleApp) の暗号化コンテキストを指定します。

```
"kms:EncryptionContext:AppName": "ExampleApp"
```

これは[単一値の条件キー](#)です。条件キーのキーは、特定の暗号化コンテキストキー (context-key) を指定します。各 API リクエストに複数の暗号化コンテキストペアを含めることができますが、指定されたコンテキストキーを持つ暗号化コンテキストペアが設定できる値は 1 つだけです。例えば、kms:EncryptionContext:Department 条件キーは、Department キーを持つ暗号化コンテキストペアにのみ適用されます。Department キーを持つ任意の暗号化コンテキストペアには 1 つの値しか設定できません。

kms:EncryptionContext:context-key 条件キーで集合演算子を使用しないでください。ポリシーステートメントを Allow アクションで作成する場合、kms:EncryptionContext:context-key 条件キー、ForAllValues 集合演算子、条件は、暗号化コンテキストのないリクエストと、ポリシー条件で指定されていない暗号化コンテキストペアを持つリクエストを許可します。

⚠ Warning

この単一値の条件キーで、`ForAnyValue` または `ForAllValues` の集合演算子を使用しないでください。これらの集合演算子は、要求する値を必要としないポリシー条件を作成し、禁止する値を許可する可能性があります。

`kms:EncryptionContext:context-key` を持つ `ForAllValues` 集合演算子を含むポリシーを作成または更新すると、は次のエラーメッセージ AWS KMS を返します。

`OverlyPermissiveCondition:EncryptionContext: Using the ForAllValues set operator with a single-valued condition key matches requests without the specified encryption context or with an unspecified encryption context. To fix, remove ForAllValues.`

特定の暗号化コンテキストペアを要求するには、`StringEquals` 演算子を持つ `kms:EncryptionContext:context-key` 条件キーを使用します。

次の例に示されるキーポリシーステートメントでは、リクエスト内の暗号化コンテキストに `AppName:ExampleApp` ペアが含まれる場合に限り、ロールを引き受けることができるプリンシパルが `GenerateDataKey` リクエストの KMS キーを使用できるようにします。他の暗号化コンテキストのペアも許可されます。

キースペース名では、大文字と小文字は区別されません。値の大文字と小文字の区別は、条件演算子 (`StringEquals` など) によって決定されます。詳細については、「[暗号化コンテキスト条件での大文字と小文字の区別](#)」を参照してください。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

暗号化コンテキストペアを要求し、他のすべての暗号化コンテキストペアを禁止するには、両方のポリシーステートメントで、`kms:EncryptionContext:context-key` および [kms:EncryptionContextKeys](#) を使用します。次のポリシーステートメントでは、`kms:EncryptionContext:AppName` 条件キーを使用してリクエストの `AppName=ExampleApp` 暗号化コンテキストペアを要求します。ForAllValues 集合演算子を持つ `kms:EncryptionContextKeys` 条件キーを使用して、`AppName` 暗号化コンテキストキーのみを許可することもできます。

ForAllValues 集合演算子は、リクエストの暗号化コンテキストキーを `AppName` に制限します。ポリシーステートメントに、ForAllValues 集合演算子を持つ `kms:EncryptionContextKeys` 条件が単独で使用された場合、この集合演算子は、暗号化コンテキストのないリクエストを許可します。ただし、リクエストに暗号化コンテキストがない場合、`kms:EncryptionContext:AppName` 条件は失敗します。ForAllValues 集合演算子の詳細については、IAM ユーザーガイドの[複数のキーと値の使用](#)を参照してください。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/KeyUsers"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    },
    "ForAllValues:StringEquals": {
      "kms:EncryptionContextKeys": [
        "AppName"
      ]
    }
  }
}
```

この条件キーを使用して、特定のオペレーションの KMS キーへのアクセスを拒否することもできます。次の例のキーポリシーステートメントでは、Deny 効果を使用して、リクエストの暗号化コンテキストに `Stage=Restricted` 暗号化コンテキストペアが含まれる場合に、プリンシパルが KMS キーを使用することを禁止します。この条件により、他の暗号化コンテキストペアを含むリクエストが許可されます。これには、`Stage` キーおよびその他の値 (`Stage=Test` など) を持つ暗号化コンテキストペアが含まれます。

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Stage": "Restricted"
    }
  }
}
```

複数の暗号化コンテキストペアを使用する

複数の暗号化コンテキストペアを要求または禁止できます。また、複数の暗号化コンテキストペアのうちの1つを要求することもできます。これらの条件を解釈するために使用されるロジックの詳細については、IAM ユーザーガイドの[複数のキーまたは値を持つ条件の作成](#)を参照してください。

Note

このトピックの以前のバージョンでは、kms:EncryptionContext:context-key 条件キーを持つ ForAnyValue および ForAllValues 集合演算子を使用するポリシーステートメントを表示していました。[単一値の条件キー](#)を持つ集合演算子を使用すると、暗号化コンテキストのないリクエストおよび暗号化コンテキストペアが指定されていないリクエストを許可するポリシーとなる可能性があります。

例えば、Allow 効果を持つポリシー条件の場合、ForAllValues 集合演算子および "kms:EncryptionContext:Department": "IT" 条件キーは、暗号化コンテキストを「Department=IT」ペアに制限しません。これは、暗号化コンテキストのないリクエストとおよび暗号化コンテキストのペアが指定されていないリクエストを許可します (Stage=Restricted など)。

ポリシーを確認し、kms:EncryptionContext:context-key を含む条件から集合演算子を削除してください。この形式のポリシーを作成または更新しようとする、OverlyPermissiveCondition の例外を含むエラーが発生します。このエラーを解決するには、集合演算子を削除します。

複数の暗号化コンテキストのペアを要求するには、同じ条件でペアを一覧表示します。次の例のキーポリシーステートメントでは、2つの暗号化コンテキストペア (Department=IT および Project=Alpha) を要求します。条件には異なるキー (kms:EncryptionContext:Department および kms:EncryptionContext:Project) が含まれるため、それらは暗黙的に AND 演算子によって接続されます。他の暗号化コンテキストペアは許可されますが、必須ではありません。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Department": "IT",
      "kms:EncryptionContext:Project": "Alpha"
    }
  }
}
```

1つの暗号化コンテキストペア、または別のペアを要求するには、各条件キーを個別のポリシーステートメントに配置します。次のキーポリシーの例では、Department=IT または Project=Alpha ペア、またはその両方を要求します。他の暗号化コンテキストペアは許可されますが、必須ではありません。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Department": "IT"
    }
  }
},
{
```

```
"Effect": "Allow",
"Principal": {
  "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
},
"Action": "kms:GenerateDataKey",
"Resource": "*",
"Condition": {
  "StringEquals": {
    "kms:EncryptionContext:Project": "Alpha"
  }
}
```

特定の暗号化ペアを要求し、他のすべての暗号化コンテキストペアを除外するには、ポリシーステートメントで、`kms:EncryptionContext:context-key` および [kms:EncryptionContextKeys](#) を使用します。次のキーポリシーステートメントでは、`kms:EncryptionContext:context-key` 条件を使用して、`Department=IT` および `Project=Alpha` ペアの両方を持つ暗号化コンテキストを要求します。このコンテキストでは、`ForAllValues` 集合演算子を持つ `kms:EncryptionContextKeys` 条件キーを使用して、`Department` および `Project` 暗号化コンテキストキーのみを許可します。

`ForAllValues` 集合演算子は、リクエストの暗号化コンテキストキーを `Department` `Project` に制限します。条件内に単独で使った場合、この集合演算子は暗号化コンテキストのないリクエストを許可しますが、この設定では、本条件の `kms:EncryptionContext:context-key` は失敗する可能性があります。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Department": "IT",
      "kms:EncryptionContext:Project": "Alpha"
    },
    "ForAllValues:StringEquals": {
      "kms:EncryptionContextKeys": [
        "Department",
        "Project"
      ]
    }
  }
}
```

```
}  
}  
}
```

複数の暗号化コンテキストペアを禁止することもできます。次の例のキーポリシーステートメントでは、Deny 効果を使用して、リクエストの暗号化コンテキストに Stage=Restricted または Stage=Production ペアが含まれる場合に、プリンシパルがKMS キーを使用することを禁止します。

同じキー (kms:EncryptionContext:Stage) の複数の値 (RestrictedおよびProduction) は、暗黙的に OR によって接続されます。詳細については、IAM ユーザーガイドの [Evaluation logic for conditions with multiple keys or values](#) を参照してください。

```
{  
  "Effect": "Deny",  
  "Principal": {  
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"  
  },  
  "Action": "kms:GenerateDataKey",  
  "Resource": "*",  
  "Condition": {  
    "StringEquals": {  
      "kms:EncryptionContext:Stage": [  
        "Restricted",  
        "Production"  
      ]  
    }  
  }  
}
```

暗号化コンテキスト条件での大文字と小文字の区別

復号オペレーションで指定される暗号化コンテキストは、暗号化オペレーションで指定される暗号化コンテキストに大文字と小文字を区別して完全に一致する必要があります。複数のペアの暗号化コンテキストのペアの順序のみを変更できます。

ただし、ポリシー条件では、条件キーの大文字と小文字は区別されません。条件値の大文字と小文字の区別は、使用する [ポリシー条件演算子](#) (StringEquals や StringEqualsIgnoreCase など) によって決まります。

したがって、kms:EncryptionContext: プレフィックスと *context-key* の置換で構成される条件キーでは、大文字と小文字は区別されません。この条件を使用するポリシーでは、条件キーのい

ずれの要素もチェックされません。値の大文字と小文字の区別 (*context-value* の置換) は、ポリシー条件演算子によって決まります。

例えば、次のポリシーステートメントでは、大文字と小文字に関係なく暗号化コンテキストに Appname キーが含まれている場合にオペレーションが許可されます。この StringEquals 条件では、指定時に ExampleApp を大文字にする必要があります。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:Appname": "ExampleApp"
    }
  }
}
```

大文字と小文字を区別する暗号化コンテキストキーを要求するには、[kms:EncryptionContextKeys](#) ポリシー条件と、大文字と小文字が区別される条件演算子 (StringEquals など) を使用します。このポリシー条件では、暗号化コンテキストキーがポリシー条件値であるため、大文字と小文字の区別は条件演算子によって決定されます。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    }
  }
}
```

暗号化コンテキストのキーと値の両方に対して大文字と小文字を区別する評価を要求するには、同じポリシーステートメントで `kms:EncryptionContextKeys` と `kms:EncryptionContext:context-key` のポリシー条件を一緒に使用します。大文字と小文字を区別する条件演算子 (`StringEquals` など) は、常に条件の値に適用されます。暗号化コンテキストキー (`AppName` など) は `kms:EncryptionContextKeys` 条件の値です。暗号化コンテキスト値 (`ExampleApp` など) は、`kms:EncryptionContext:context-key` 条件の値です。

例えば、次のキーポリシーステートメントでは、`StringEquals` 演算子で大文字と小文字が区別されるため、暗号化コンテキストキーと暗号化コンテキスト値の両方で大文字と小文字が区別されます。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    },
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

暗号化コンテキスト条件での変数の使用

暗号化コンテキストペアのキーと値はシンプルなリテラル文字列であることが必要です。整数やオブジェクトなど完全に解決されない型のキーと値は使用できません。整数や浮動小数点数など、別の型を使用する場合、はそれをリテラル文字列として AWS KMS 解釈します。

```
"encryptionContext": {
  "department": "10103.0"
}
```

ただし、`kms:EncryptionContext:context-key` 条件キーの値は [IAM ポリシー変数](#) となる可能性があります。これらのポリシー変数はリクエストの値に基づいて実行時に解決されます。例え

ば、`aws:CurrentTime` はリクエストの時間に解決され、`aws:username` は呼び出し元のフレンドリ名に解決されます。

これらのポリシー変数を使用し、ポリシーステートメントを作成して、暗号化コンテキストに限定的な情報 (呼び出し元のユーザー名など) を必要とする条件を指定できます。ポリシーステートメントに変数を含めるため、ロールを引き受けることができるすべてのユーザーに同じポリシーステートメントを使用できます。ユーザー別にポリシーステートメントを記述する必要はありません。

ロールを引き受けることができるすべてのユーザーが同じ KMS キーを使用して、データを暗号化および復号する状況を考慮します。ただし、それらのユーザーに自分が暗号化したデータのみの復号を許可するとします。まず、すべてのリクエストに、キーが `user` で値が発信者の AWS ユーザー名である暗号化コンテキスト AWS KMS を含めるように要求します。次に例を示します。

```
"encryptionContext": {
  "user": "bob"
}
```

次に、この要件を定義するために、以下の例のようなポリシーステートメントを使用できます。このポリシーステートメントでは、`TestTeam` ロールに、KMS キーを使用してデータを暗号化および復号するためのアクセス許可を付与します。ただし、そのアクセス許可は、`"user": "<username>"` ペアがリクエストの暗号化コンテキストに含まれる場合にのみ有効です。条件では、ユーザー名を表すために [aws:username](#) policy 変数を使用します。

リクエストが評価されるとき、条件の変数が呼び出し元のユーザー名に置き換えられます。そのため条件では、「bob」には `"user": "bob"`、「alice」には `"user": "alice"` の暗号化コンテキストを必須とします。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/TestTeam"
  },
  "Action": [
    "kms:Decrypt",
    "kms:Encrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:user": "${aws:username}"
    }
  }
}
```

```
}  
}  
}
```

IAM ポリシー変数は、`kms:EncryptionContext:context-key` 条件キーの値でのみ使用できます。キーで変数を使用することはできません。

変数に [プロバイダ固有のコンテキストキー](#) を使用することもできます。これらのコンテキストキー AWS は、ウェブ ID フェデレーションを使用して にログインしたユーザーを一意に識別します。

すべての変数と同様に、これらの変数は、実際の暗号化コンテキストではなく `kms:EncryptionContext:context-key` ポリシー条件でのみ使用できます。また、条件のキーではなく値でのみ使用できます。

例えば、以下のキーポリシーステートメントは前のものと似ています。ただし、この条件には、Amazon Cognito ユーザープールにログインしたユーザーを値が一意に識別する、キーが `sub` の暗号化コンテキストが必要です。Amazon Cognito でのユーザーおよびロールの識別についての詳細は、[Amazon Cognito デベロッパーガイド](#) の [IAM ロール](#) を参照してください。

```
{  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::111122223333:role/TestTeam"  
  },  
  "Action": [  
    "kms:Decrypt",  
    "kms:Encrypt"  
  ],  
  "Resource": "*",  
  "Condition": {  
    "StringEquals": {  
      "kms:EncryptionContext:sub": "${cognito-identity.amazonaws.com:sub}"  
    }  
  }  
}
```

以下の資料も参照してください。

- [the section called “kms:EncryptionContextKeys”](#)
- [the section called “kms:GrantConstraintType”](#)

kms:EncryptionContextKeys

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Encry ptionCont extKeys	文字列 (リスト)	複数値	CreateGrant	キーポリシーと IAM ポリシー
			Decrypt	
			Encrypt	
			Generated ataKey	
			Generated ataKeyPair	
			Generated ataKeyPai rWithoutP laintext	
			Generated ataKeyWit houtPlain text	
			ReEncrypt	
			RetireGrant	

kms:EncryptionContextKeys 条件キーを使用することで、暗号化オペレーションに対するリクエスト内の[暗号化コンテキスト](#)に基づいて[対称暗号化 KMS キー](#)へのアクセスを制御することができます。各暗号化コンテキストペアのキーのみを評価するには、この条件キーを使用します。暗号化コンテキストのキーと値の両方を評価するには、kms:EncryptionContext:context-key 条件キーを使用します。

[非対称 KMS キー](#)または [HMAC KMS キー](#)を使用する暗号化オペレーションで暗号化コンテキストを指定することはできません。非対称アルゴリズムと MAC アルゴリズムは、暗号化コンテキストをサポートしません。

 Note

暗号化コンテキストキーを含む条件キー値は、AWS KMS キーポリシーの文字とエンコーディングルールに準拠する必要があります。この条件キーは、すべての有効な暗号化コンテキストキーを表現するために使用できない場合があります。キーポリシードキュメントのルールに関する詳細については、「[キーポリシー形式](#)」を参照してください。IAM ポリシードキュメントのルールに関する詳細については、「IAM ユーザーガイド」の「[IAM 名前の要件](#)」を参照してください。

これは[複数値の条件キー](#)です。各 API リクエストで複数の暗号化コンテキストペアを指定できます。kms:EncryptionContextKeys は、リクエストの暗号化コンテキストキーとポリシーの暗号化コンテキストキーのセットを比較します。これらのセットを比較する方法を決定するには、ForAnyValue または ForAllValues 集合演算子をポリシー条件で使います。集合演算子の詳細については、IAM ユーザーガイドの[複数のキーと値の使用](#)を参照してください。

- ForAnyValue: リクエスト内の 1 つ以上の暗号化コンテキストキーがポリシー条件の暗号化コンテキストキーと一致する必要があります。その他の暗号化コンテキストキーも許可されます。リクエストに暗号化コンテキストがない場合、条件は満たされません。
- ForAllValues: リクエスト内のすべての暗号化コンテキストキーがポリシー条件の暗号化コンテキストキーと一致する必要があります。この集合演算子は、暗号化コンテキストキーをポリシー条件内のキーに制限します。暗号化コンテキストキーは必要ありませんが、指定されていない暗号化コンテキストキーは禁止されています。

次の例のキーポリシーステートメントでは、ForAnyValue 集合演算子で

kms:EncryptionContextKeys 条件キーを使います。このポリシーステートメントでは KMS キーを使用して、リクエストの暗号化コンテキストペアの 1 つ以上に、値にかかわらず AppName キーが含まれる場合にのみ、指定されたオペレーションで KMS キーの使用を許可します。

例えば、このキーポリシーステートメントでは、2 つの暗号化コンテキストペア AppName=Helper および Project=Alpha を持つ GenerateDataKey リクエストを許可します。これは、最初の暗号化コンテキストペアが条件を満たすためです。Project=Alpha のみを持つリクエスト、または暗号化コンテキストがないリクエストは失敗します。

[StringEquals](#) 条件オペレーションでは大文字と小文字が区別されるため、このポリシーステートメントでは、暗号化コンテキストキーのスペル、および大文字と小文字が区別されます。ただし、キーの大文字と小文字の区別を無視する条件演算子 ([StringEqualsIgnoreCase](#) など) を使用できます。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": [
    "kms:Encrypt",
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:EncryptionContextKeys": "AppName"
    }
  }
}
```

`kms:EncryptionContextKeys` 条件キーを使用して、KMS キーを使用する暗号化オペレーションで、暗号化テキスト (任意の暗号化コンテキスト) を要求することもできます。

次の例のキーポリシーステートメントでは、`kms:EncryptionContextKeys` 条件キーを [Null 条件演算子](#) とともに使用して、API リクエストに暗号化コンテキストが存在する (null ではない) 場合にのみ、KMS キーへのアクセスを許可します。この条件では、暗号化コンテキストのキーまたは値をチェックしません。暗号化コンテキストが存在することだけを検証します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": [
    "kms:Encrypt",
    "kms:GenerateDataKey*"
  ],
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:EncryptionContextKeys": false
    }
  }
}
```

```
}  
}  
}
```

以下の資料も参照してください。

- [kms:EncryptionContext: context-key](#)
- [kms:GrantConstraintType](#)

kms:ExpirationModel

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Expir ationModel	String	単一値	ImportKey Material	キーポリシーと IAM ポリシー

kms:ExpirationModel 条件キーは、リクエストの [ExpirationModel](#) パラメータの値に基づき、[ImportKeyMaterial](#) オペレーションへのアクセスを制御します。

ExpirationModel は、インポートされたキーマテリアルの有効期限が切れているかどうかを判断するオプションのパラメータです。有効な値は、KEY_MATERIAL_EXPIRES および KEY_MATERIAL_DOES_NOT_EXPIRE です。KEY_MATERIAL_EXPIRES はデフォルト値です。

有効期限の日付と時刻は、[Validto](#) パラメーターの値によって決まります。ValidTo パラメータの値が ExpirationModel である場合を除き、KEY_MATERIAL_DOES_NOT_EXPIRE パラメータが必要です。また、[kms:ValidTo](#) 条件キーを使用して、アクセス条件として特定の有効期限を設定することもできます。

次のポリシーステートメントの例では、kms:ExpirationModel 条件キーを使用して、リクエストに ExpirationModel パラメータが含まれていて、その値が KEY_MATERIAL_DOES_NOT_EXPIRE の場合にのみ、ユーザーが KMS キーにキーマテリアルをインポートできるようにします。

```
{  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
```

```
    },
    "Action": "kms:ImportKeyMaterial",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:ExpirationModel": "KEY_MATERIAL_DOES_NOT_EXPIRE"
      }
    }
  }
}
```

また、kms:ExpirationModel 条件キーを使用して、キーマテリアルの有効期限が切れている場合にのみ、ユーザーがキーマテリアルをインポートできるようにもします。次のポリシーステートメントの例では、[Null 条件演算子](#)とともに kms:ExpirationModel 条件キーを使用して、リクエストに ExpirationModel パラメータが含まれていない場合にのみ、ユーザーがキーマテリアルをインポートできるようにします。ExpirationModel のデフォルト値は KEY_MATERIAL_EXPIRES です。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:ImportKeyMaterial",
  "Resource": "*",
  "Condition": {
    "Null": {
      "kms:ExpirationModel": true
    }
  }
}
```

以下の資料も参照してください。

- [kms:ValidTo](#)
- [kms:WrappingAlgorithm](#)
- [kms:WrappingKeySpec](#)

kms:GrantConstraintType

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:GrantConstraintType	String	単一値	CreateGrant RetireGrant	キーポリシーと IAM ポリシー

この条件キーを使用し、リクエストの[権限の制約](#)のタイプに基づいて、[CreateGrant](#) オペレーションへのアクセスを制御できます。

許可の作成では、オプションで許可の制約を指定して、特定の[暗号化コンテキスト](#)が存在する場合のみ、許可によってオペレーションを実行できます。許可の制約には、EncryptionContextEquals または EncryptionContextSubset の 2 つのタイプがあります。この条件キーを使用して、リクエストにどちらのタイプが含まれているか確認できます。

Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

次のキーポリシーステートメントの例では、kms:GrantConstraintType 条件キーを使用して、リクエストに EncryptionContextEquals 権限の制約が含まれている場合にのみ、ユーザーが権限を作成できるようにします。この例は、キーポリシーのポリシーステートメントを示しています。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:GrantConstraintType": "EncryptionContextEquals"
    }
  }
}
```



```
}
}
```

以下の資料も参照してください。

- [kms:EncryptionContext: context-key](#)
- [kms:EncryptionContextKeys](#)
- [kms:GrantIsForAWSResource](#)
- [kms:GrantOperations](#)
- [kms:GranteePrincipal](#)
- [kms:RetiringPrincipal](#)

kms:GrantIsForAWSResource

AWS KMS 条件 キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:GrantIsForAWSResource	ブール値	単一値	CreateGrant ListGrants RevokeGrant	キーポリシーと IAM ポリシー

[AWS KMSと統合されたAWS のサービス](#)がユーザーの代わりにオペレーションを呼び出す場合にのみ、[CreateGrant](#)、[ListGrants](#)、もしくは[RevokeGrant](#) のオペレーション向けアクセス許可を付与または拒否します。このポリシー条件では、ユーザーがこれらの許可オペレーションを直接呼び出すことはできません。

次のキーポリシーステートメント例では、kms:GrantIsForAWSResource 条件キーを使用しています。これにより AWS KMS、Amazon EBS などのと統合されている AWS サービスが、指定されたプリンシパルに代わってこの KMS キーに許可を作成できるようになります。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
}
```

```

{
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:GrantIsForAWSResource": true
    }
  }
}

```

以下の資料も参照してください。

- [kms:GrantConstraintType](#)
- [kms:GrantOperations](#)
- [kms:GranteePrincipal](#)
- [kms:RetiringPrincipal](#)

kms:GrantOperations

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Grant Operations	String	複数値	CreateGrant	キーポリシーと IAM ポリシー

この条件キーを使用し、リクエストの[権限オペレーション](#)に基づいて、[CreateGrant](#) オペレーションへのアクセスを制御できます。例えば、暗号化へのアクセス権限を委任し、復号化へのアクセス権限を委任しないという許可をユーザーが作成するようにできます。権限の詳細については、[権限の使用](#)を参照してください。

これは[複数値を持つ条件キー](#)です。kms:GrantOperations は CreateGrant リクエストの権限オペレーションのセットをポリシーの権限オペレーションのセットと比較します。これらのセットを比較する方法を決定するには、ForAnyValue または ForAllValues 集合演算子をポリシー条件で使用します。集合演算子の詳細については、IAM ユーザーガイドの[複数のキーと値の使用](#)を参照してください。

- ForAnyValue: リクエストの 1 つ以上の権限オペレーションが、ポリシー条件の権限オペレーションのうちの 1 つと一致する必要があります。その他の権限オペレーションは許可されます。

- **ForAllValues:** リクエスト内のすべての権限オペレーションが、ポリシー条件の権限オペレーションと一致する必要があります。この集合演算子は、権限オペレーションをポリシー条件で指定されたオペレーションに制限します。この集合演算子は権限オペレーションを必要としませんが、不特定の権限オペレーションを禁止します。

ForAllValues はリクエストに権限オペレーションがない場合にも **true** を返しますが、**CreateGrant** はそれを許可しません。Operations パラメータが欠落しているか、**null** 値を持っている場合、**CreateGrant** リクエストは失敗します。

次のキーポリーステートメントの例では、**kms:GrantOperations** 条件キーを使用して、権限オペレーションが **Encrypt**、**ReEncryptTo**、または両方の場合にのみ、権限を作成します。権限に他のオペレーションが含まれている場合、**CreateGrant** リクエストは失敗します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "ForAllValues:StringEquals": {
      "kms:GrantOperations": [
        "Encrypt",
        "ReEncryptTo"
      ]
    }
  }
}
```

ポリシー条件で集合演算子を **ForAnyValue** に変更した場合、ポリーステートメントでは、権限の権限オペレーションの 1 つ以上が **Encrypt** または **ReEncryptTo** であっても、**Decrypt** または **ReEncryptFrom** のような他の権限オペレーションを許可する必要があります。

以下の資料も参照してください。

- [kms:GrantConstraintType](#)
- [kms:GrantIsForAWSResource](#)
- [kms:GranteePrincipal](#)

- [kms:RetiringPrincipal](#)

kms:GranteePrincipal

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:GranteePrincipal	String	単一値	CreateGrant	IAM とキーポリ シー

この条件キーを使用して、リクエストの [CreateGrant](#) パラメータの値に基づき、[GranteePrincipal](#) オペレーションへのアクセスを制御できます。例えば、CreateGrant リクエストの被付与者プリンシパルが条件ステートメントで指定されたプリンシパルと一致した場合にのみ、KMS キーを使用する権限を作成できます。

グラント対象のプリンシパルを指定するには、AWS プリンシパルの Amazon リソースネーム (ARN) を使用します。有効なプリンシパルには AWS アカウント、IAM ユーザー、IAM ロール、フェデレーティッドユーザー、引き受けたロールユーザーが含まれます。プリンシパル向け ARN 構文については、「IAM ユーザーガイド」の「[IAM の ARN](#)」を参照してください。

次のキーポリーステートメントの例では、kms:GranteePrincipal 条件キーを使用して、権限の被付与者プリンシパルが LimitedAdminRole の場合にのみ、KMS キーの権限を作成します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:GranteePrincipal": "arn:aws:iam::111122223333:role/LimitedAdminRole"
    }
  }
}
```

以下の資料も参照してください。

- [kms:GrantConstraintType](#)
- [kms:GrantIsForAWSResource](#)
- [kms:GrantOperations](#)
- [kms:RetiringPrincipal](#)

kms:KeyAgreementAlgorithm

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:KeyAgreementAlgorithm	String	単一値	DeriveSharedSecret	キーポリシーと IAM ポリシー

kms:KeyAgreementAlgorithm 条件キーを使用することで、リクエスト内の KeyAgreementAlgorithm パラメータの値に基づいて [DeriveSharedSecret](#) オペレーションへのアクセスを制御できます。KeyAgreementAlgorithm では、唯一の有効な値は ECDH です。

例えば、次のキーポリシーステートメントは、kms:KeyAgreementAlgorithm 条件キーを使用して、KeyAgreementAlgorithm が ECDH でない限り、DeriveSharedSecret へのすべてのアクセスを拒否します。

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:DeriveSharedSecret",
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "kms:KeyAgreementAlgorithm": "ECDH"
    }
  }
}
```

以下の資料も参照してください。

- [the section called “kms:KeyUsage”](#)

kms:KeyOrigin

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:KeyOrigin	String	単一値	CreateKey KMS キーリソースのオペレーション	IAM ポリシー キーポリシーと IAM ポリシー

kms:KeyOrigin 条件キーは、オペレーションによって作成される、またはオペレーションで 사용되는 KMS キーの Origin プロパティの値に基づいて、オペレーションへのアクセスを制御します。これは、リソース条件または要求条件として機能します。

この条件キーを使用して、リクエストの [Origin](#) パラメータの値に基づき、[CreateKey](#) オペレーションへのアクセスを制御できます。Origin の有効値は、AWS_KMS、AWS_CLOUDHSM、および EXTERNAL です。

たとえば、キーマテリアルが AWS KMS (AWS_KMS) で生成されたとき、キーマテリアルが [カスタムキーストア](#) () に関連付けられた AWS CloudHSM クラスターで生成されたとき、またはキーマテリアルが外部ソース (AWS_CLOUDHSM) からインポートされたときのみ、KMS キーを作成できます EXTERNAL。 [???](#)

次のキーポリシーステートメントの例では、に kms:KeyOrigin 条件キーを使用して、がキーマテリアルを作成する場合にのみ KMS キー AWS KMS を作成します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
      }
    }
  ]
}
```

```
    },
    "Action": "kms:CreateKey",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:KeyOrigin": "AWS_KMS"
      }
    }
  },
  {
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
    },
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:GenerateDataKey",
      "kms:GenerateDataKeyWithoutPlaintext",
      "kms:GenerateDataKeyPair",
      "kms:GenerateDataKeyPairWithoutPlaintext",
      "kms:ReEncrypt*"
    ],
    "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
    "Condition": {
      "StringEquals": {
        "kms:KeyOrigin": "AWS_CLOUDHSM"
      }
    }
  }
]
```

kms:KeyOrigin 条件キーを使用し、オペレーションに使用される KMS キーの Origin プロパティに基づいて、KMS キーを使用または管理するオペレーションへのアクセスを制御することもできます。このオペレーションは KMS キーリソースオペレーションである必要があります。つまり、特定の KMS キーに認可されるオペレーションです。KMS キーリソースオペレーションを識別するには、[アクションとリソースの表](#)で、オペレーションの Resources 列の KMS key の値を探します。

例えば次の IAM ポリシーでは、カスタムキーストアで作成されたアカウントの KMS キーのみを使用して、プリンシパルが、指定された KMS キーリソースのオペレーションを実行できるようにします。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:GenerateDataKey",
    "kms:GenerateDataKeyWithoutPlaintext",
    "kms:GenerateDataKeyPair",
    "kms:GenerateDataKeyPairWithoutPlaintext",
    "kms:ReEncrypt*"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "AWS_CLOUDHSM"
    }
  }
}
```

以下の資料も参照してください。

- [kms:BypassPolicyLockoutSafetyCheck](#)
- [kms:KeySpec](#)
- [kms:KeyUsage](#)

kms:KeySpec

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:KeySpec	String	単一値	CreateKey	IAM ポリシー
			KMS キーリソー スのオペレー ション	キーポリシーと IAM ポリシー

kms:KeySpec 条件キーは、オペレーションによって作成される、またはオペレーションで使用される KMS キーの KeySpec プロパティの値に基づいて、オペレーションへのアクセスを制御します。

IAM ポリシーにあるこの条件キーを使用し、CreateKey リクエストの [KeySpec](#) パラメータ値に基づいて、[CreateKey](#) オペレーションへのアクセスを制御できます。例えば、この条件を使用して、対称暗号化 KMS キーのみの作成、または HMAC KMS キーのみの作成をユーザーに許可することができます。

以下の IAM ポリシーステートメント例は、kms:KeySpec 条件キーを使用して、RSA 非対称 KMS キーのみの作成をプリンシパルに許可します。この許可は、リクエスト内の KeySpec が RSA_ で始まる場合に限り、有効です。

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:KeySpec": "RSA_*"
    }
  }
}
```

kms:KeySpec 条件キーを使用し、オペレーションに使用される KMS キーの KeySpec プロパティに基づいて、KMS キーを使用または管理するオペレーションへのアクセスを制御することもできます。このオペレーションは KMS キーリソースオペレーションである必要があります。つまり、特定の KMS キーに認可されるオペレーションです。KMS キーリソースオペレーションを識別するには、[アクションとリソースの表](#)で、オペレーションの Resources 列の KMS key の値を探します。

例えば、以下の IAM ポリシーは、指定された KMS キーリソースオペレーションの実行をプリンシパルに許可しますが、許可されるのはアカウント内の対称暗号化 KMS キーを使用する場合のみになります。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:DescribeKey"
  ],
```

```
"Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
"Condition": {
  "StringEquals": {
    "kms:KeySpec": "SYMMETRIC_DEFAULT"
  }
}
```

以下の資料も参照してください。

- [kms:BypassPolicyLockoutSafetyCheck](#)
- [kms:CustomerMasterKeySpec](#) (非推奨)
- [kms:DataKeyPairSpec](#)
- [kms:KeyOrigin](#)
- [kms:KeyUsage](#)

kms:KeyUsage

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:KeyUs age	String	単一値	CreateKey	IAM ポリシー
			KMS キーリソー スのオペレー ション	キーポリシーと IAM ポリシー

kms:KeyUsage 条件キーは、オペレーションによって作成される、またはオペレーションで使用される KMS キーの KeyUsage プロパティの値に基づいて、オペレーションへのアクセスを制御します。

この条件キーを使用し、リクエストの [KeyUsage](#) パラメータの値に基づいて、[CreateKey](#) オペレーションへのアクセスを制御できます。KeyUsage の有効値は、ENCRYPT_DECRYPT、SIGN_VERIFY、GENERATE_VERIFY_MAC、KEY_AGREEMENT です。

例えば、KeyUsage が ENCRYPT_DECRYPT である場合にのみ KMS キーを作成し、KeyUsage が SIGN_VERIFY の場合はユーザーのアクセス許可を拒否できます。

次の IAM ポリシーステートメントの例では、`kms:KeyUsage` 条件キーを使用して、`KeyUsage` が `ENCRYPT_DECRYPT` の場合にのみ KMS キーを作成します。

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:KeyUsage": "ENCRYPT_DECRYPT"
    }
  }
}
```

`kms:KeyUsage` 条件キーを使用し、オペレーションの KMS キーの `KeyUsage` プロパティに基づいて、KMS キーを使用または管理するオペレーションへのアクセスを制御することもできます。このオペレーションは KMS キーリソースオペレーションである必要があります。つまり、特定の KMS キーに認可されるオペレーションです。KMS キーリソースオペレーションを識別するには、[アクションとリソースの表](#)で、オペレーションの `Resources` 列の KMS key の値を探します。

例えば次の IAM ポリシーでは、署名と検証に使用されるアカウント内の KMS キーのみを使用して、指定された KMS キーリソースのオペレーションをプリンシパルが実行できるようにします。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:CreateGrant",
    "kms:DescribeKey",
    "kms:GetPublicKey",
    "kms:ScheduleKeyDeletion"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyUsage": "SIGN_VERIFY"
    }
  }
}
```

以下の資料も参照してください。

- [kms:BypassPolicyLockoutSafetyCheck](#)

- [kms:CustomerMasterKeyUsage](#) (非推奨)
- [kms:KeyOrigin](#)
- [kms:KeySpec](#)

kms:MacAlgorithm

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:MacAlgorithm	String	単一値	GenerateMac VerifyMac	キーポリシーと IAM ポリシー

kms:MacAlgorithm 条件キーを使用することで、リクエスト内の MacAlgorithm パラメータの値に基づいて [GenerateMac](#) および [VerifyMac](#) オペレーションへのアクセスを制御することができます。

以下のキーポリシー例は、リクエスト内の MAC アルゴリズムが HMAC_SHA_384 または HMAC_SHA_512 である場合に限り、HMAC KMS キーを使用して HMAC タグを生成し、検証することを、testers ロールを引き受けることができるユーザーに許可します。このポリシーは、それぞれが独自の条件を持つ 2 つの個別のポリシーステートメントを使用します。単一の条件ステートメント内で複数の MAC アルゴリズムを指定する場合、条件はアルゴリズムのどちらか一方ではなく、両方を必須とします。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/testers"
      },
      "Action": [
        "kms:GenerateMac",
        "kms:VerifyMac"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:MacAlgorithm": "HMAC_SHA_384"
      }
    }
  },
  {
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/testers"
    },
    "Action": [
      "kms:GenerateMac",
      "kms:VerifyMac"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:MacAlgorithm": "HMAC_SHA_512"
      }
    }
  }
]
}

```

以下の資料も参照してください。

- [the section called “kms:EncryptionAlgorithm”](#)
- [kms:SigningAlgorithm](#)

kms:MessageType

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Messa geType	String	単一値	Sign Verify	キーポリシーと IAM ポリシー

kms:MessageType 条件キーは、リクエストの MessageType パラメータの値に基づいて、[Sign](#) および [Verify](#) オペレーションへのアクセスを制御します。MessageType の有効値は、RAW と DIGEST です。

例えば、次のキーポリシーステートメントでは kms:MessageType 条件キーを使用して、非対称 KMS キーを使用してメッセージに署名することを許可し、メッセージダイジェストの使用は拒否しません。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:Sign",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:MessageType": "RAW"
    }
  }
}
```

以下の資料も参照してください。

- [the section called “kms:SigningAlgorithm”](#)

kms:MultiRegion

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:MultiRegion	ブール値	単一値	CreateKey KMS キーリソースのオペレーション	キーポリシーと IAM ポリシー

この条件キーを使用すると、単一リージョンキー、または[マルチリージョンキー](#)のみのオペレーションを許可できます。kms:MultiRegion 条件キーは、KMS キーの MultiRegion プロパティの値に

基づいて、KMS キーに対する AWS KMS オペレーションと [CreateKey](#) オペレーションへのアクセスを制御します。有効な値は、true (マルチリージョン)、および false (単一リージョン) です。すべての KMS キーには MultiRegion プロパティが含まれます。

例えば、次の IAM ポリシーステートメントでは kms:MultiRegion 条件キーを使用して、プリンシパルが単一リージョンキーを作成できるようにします。

```
{
  "Effect": "Allow",
  "Action": "kms:CreateKey",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms:MultiRegion": false
    }
  }
}
```

kms:MultiRegionKeyType

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:MultiRegionKeyType	String	単一値	CreateKey KMS キーリソースのオペレーション	キーポリシーと IAM ポリシー

この条件キーを使用して、[マルチリージョンプライマリキー](#)または[マルチリージョンレプリカキー](#)のみのオペレーションを許可できます。kms:MultiRegionKeyType 条件キーは、KMS キーの MultiRegionKeyType プロパティに基づいて、KMS キーに対する AWS KMS オペレーションと [CreateKey](#) オペレーションへのアクセスを制御します。有効な値は PRIMARY および REPLICa です。マルチリージョンキーのみに MultiRegionKeyType プロパティがあります

通常は、IAM ポリシーの kms:MultiRegionKeyType 条件キーを使用して、複数の KMS キーへのアクセスを制御します。ただし、特定のマルチリージョンキーがプライマリまたはレプリカに変更されることがあるため、キーポリシーでこの条件を使用して、特定のマルチリージョンキーがプライマリキーまたはレプリカキーである場合にのみ、オペレーションを許可します。

例えば、次の IAM ポリシーステートメントでは `kms:MultiRegionKeyType` 条件キーを使用して、指定した AWS アカウントのマルチリージョンレプリカキーのみでキー削除のスケジュールおよびキャンセルを実行することをプリンシパルに許可します。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "arn:aws:kms:*:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:MultiRegionKeyType": "REPLICA"
    }
  }
}
```

すべてのマルチリージョンキーへのアクセスを許可または拒否するには、両方の値または `kms:MultiRegionKeyType` の null 値を使用します。ただし、この目的には [kms:MultiRegion](#) 条件キーの使用を推奨します。

kms:PrimaryRegion

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
<code>kms:PrimaryRegion</code>	文字列 (リスト)	単一値	<code>UpdatePrimaryRegion</code>	キーポリシーと IAM ポリシー

この条件キーを使用して、[UpdatePrimaryRegion](#) オペレーションの送信先リージョンを制限します。これらは AWS リージョン、マルチリージョンのプライマリキーをホストできる です。

`kms:PrimaryRegion` 条件キーは、`PrimaryRegion` パラメータ値に基づいて [UpdatePrimaryRegion](#) オペレーションへのアクセスを制御します。`PrimaryRegion` パラメータは、プライマリに昇格するマルチリージョンレプリカキー AWS リージョン のを指定します。
[???条件](#)の値は、`us-east-1` などの 1 つ以上の AWS リージョン 名前 `ap-southeast-2`、または `eu-*` などのリージョン名パターンです。

例えば、次のキーポリシーステートメントでは `kms:PrimaryRegion` 条件キーを使用して、プリンシパルがマルチリージョンキーのプライマリリージョンを、指定した 4 つのリージョンのうちの 1 つに更新できるようにします。

```
{
  "Effect": "Allow",
  "Action": "kms:UpdatePrimaryRegion",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Developer"
  },
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:PrimaryRegion": [
        "us-east-1",
        "us-west-2",
        "eu-west-3",
        "ap-southeast-2"
      ]
    }
  }
}
```

kms:ReEncryptOnSameKey

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータ입
<code>kms:ReEncryptOnSameKey</code>	ブール値	単一値	ReEncrypt	キーポリシーと IAM ポリシー

この条件キーを使用し、元の 暗号化に使用されたものと同じ送信先 KMS キーがリクエストで指定されているかどうかに基づいて、[ReEncrypt](#) オペレーションへのアクセスを制御できます。

例えば、次のポリシーステートメントでは `kms:ReEncryptOnSameKey` 条件キーを使用して、対象の KMS キーが元の暗号化に使用されたものと同じである場合にのみ再暗号化します。

```
{
  "Effect": "Allow",
```

```

"Principal": {
  "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
},
"Action": "kms:ReEncrypt*",
"Resource": "*",
"Condition": {
  "Bool": {
    "kms:ReEncryptOnSameKey": true
  }
}
}

```

kms:RequestAlias

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:RequestAlias	文字列 (リスト)	単一値	暗号化オペレーション DescribeKey GetPublicKey	キーポリシーと IAM ポリシー

この条件キーを使用して、リクエストが KMS キーを識別するために特定のエイリアスを使用する場合にのみ、オペレーションを許可します。kms:RequestAlias 条件キーでは、リクエスト内の KMS キーを識別する[エイリアス](#)に基づいて、暗号化オペレーション (GetPublicKey または DescribeKey) で使用される KMS キーへのアクセスを制御します。(このポリシー条件は、[GenerateRandom](#) オペレーションが KMS キーまたはエイリアスを使用しないため、オペレーションには影響しません)。

この条件は、[属性ベースのアクセスコントロール](#) (ABAC) をサポートします。これにより AWS KMS、KMS キーのタグとエイリアスに基づいて KMS キーへのアクセスを制御できます。ポリシーや権限を変更せずに、タグとエイリアスを使用して KMS キーへのアクセスを許可または拒否できます。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

このポリシー条件でエイリアスを指定するには、[エイリアス名](#) (alias/project-alpha など)、またはエイリアス名パターン (alias/*test* など) を使用します。この条件キーの値に[エイリアス ARN](#)を指定することはできません。

この条件を満たすには、リクエストの `KeyId` パラメータの値が、一致するエイリアス名またはエイリアス ARN である必要があります。リクエストが別の [キー識別子](#) を使用する場合、同じ KMS キーを識別しても条件を満たしません。

例えば、次のキーポリシーステートメントでは、プリンシパルが KMS キーで [GenerateDataKey](#) オペレーションを呼び出すことを許可します。ただし、これは、リクエスト内の `KeyId` パラメータ値が `alias/finance-key`、またはそのエイリアス名を持つエイリアス ARN (`arn:aws:kms:us-west-2:111122223333:alias/finance-key` など)である場合にのみ許可されます。

```
{
  "Sid": "Key policy using a request alias condition",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/developer"
  },
  "Action": "kms:GenerateDataKey",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:RequestAlias": "alias/finance-key"
    }
  }
}
```

この条件キーを使用して、エイリアスオペレーションへのアクセスを制御することはできません ([CreateAlias](#) または [DeleteAlias](#) など)。エイリアスオペレーションへのアクセスの制御については、[エイリアスへのアクセスの制御](#) を参照してください。

kms:ResourceAliases

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:ResourceAliases	文字列 (リスト)	複数値	KMS キーリソースのオペレーション	IAM ポリシーのみ

この条件キーを使用し、KMS キーに関連付けられた [エイリアス](#) に基づいて、KMS キーへのアクセスを制御します。このオペレーションは KMS キーリソースオペレーションである必要があります。つ

まり、特定の KMS キーに認可されるオペレーションです。KMS キーリソースオペレーションを識別するには、[アクションとリソースの表](#)で、オペレーションの Resources 列の KMS key の値を探します。

この条件では、AWS KMSの属性ベースのアクセスコントロール (ABAC) をサポートします。ABAC を使用すると、KMS キーに割り当てられたタグと KMS キーに関連付けられたエイリアスに基づいて、KMS キーへのアクセスを制御できます。ポリシーや権限を変更せずに、タグとエイリアスを使用して KMS キーへのアクセスを許可または拒否できます。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

エイリアスは AWS アカウント および リージョンで一意である必要がありますが、この条件により、同じリージョン内の複数の KMS キー (StringLike比較演算子を使用) または AWS リージョン各アカウントの異なる複数の KMS キーへのアクセスを制御できます。

Note

[kms:ResourceAliases](#) 条件は、KMS キーが [aliases per KMS key](#) クォータに適合している場合にのみ有効です。KMS キーがこのクォータを超えると、KMS キーを [kms:ResourceAliases](#) 条件で使用するよう認可されたプリンシパルは、KMS キーへのアクセスを拒否されます。

このポリシー条件でエイリアスを指定するには、[エイリアス名](#) (alias/project-alpha など)、またはエイリアス名パターン (alias/*test* など) を使用します。この条件キーの値に[エイリアス ARN](#)を指定することはできません。条件を満たすには、オペレーションで使用する KMS キーが指定されたエイリアスを持っている必要があります。オペレーションのリクエストで KMS キーが識別されるかどうか、またはどのように識別されるかは関係ありません。

これは、KMS キーに関連付けられたエイリアスのセットとポリシー内のエイリアスのセットを比較する、複数値を持つ条件キーです。これらのセットを比較する方法を決定するには、ForAnyValue または ForAllValues 集合演算子をポリシー条件で使用します。集合演算子の詳細については、IAM ユーザーガイドの[複数のキーと値の使用](#)を参照してください。

- ForAnyValue: KMS キーに関連付けられた 1 つ以上のエイリアスがポリシー条件のエイリアスと一致する必要があります。その他のエイリアスは許可されます。KMS キーにエイリアスがない場合、条件は満たされません。
- ForAllValues: KMS キーに関連付けられたすべてのエイリアスがポリシーのエイリアスと一致する必要があります。この集合演算子は、KMS キーと関連付けられるエイリアスを、ポリシー条件内のエイリアスに制限します。エイリアスを必要としませんが、不特定のエイリアスを禁止します。

たとえば、次の IAM ポリシーステートメントでは、プリンシパルがエイリアスに関連付けられている指定された の任意の KMS キーに対して [GenerateDataKey](#) `finance-key` オペレーションを呼び出す AWS アカウント ことを許可します。(影響を受ける KMS キーのキーポリシーでは、プリンシパルのアカウントに、このオペレーションでキーを使用することも許可する必要があります)。KMS キーに関連付けられる可能性がある多数のエイリアスのいずれかが `alias/finance-key` である場合に、条件が満たされていることを示すには条件に `ForAnyValue` 集合演算子を使用します。

`kms:ResourceAliases` 条件はリクエストではなくリソースに基づいているため、`GenerateDataKey` に対する呼び出しは、リクエストが KMS キーを識別するために [キー ID](#) または [キー ARN](#) を使用している場合でも、`finance-key` エイリアスに関連付けられているすべての KMS キーに対して成功します。

```
{
  "Sid": "AliasBasedIAMPolicy",
  "Effect": "Allow",
  "Action": "kms:GenerateDataKey",
  "Resource": [
    "arn:aws:kms:*:111122223333:key/*",
    "arn:aws:kms:*:444455556666:key/*"
  ],
  "Condition": {
    "ForAnyValue:StringEquals": {
      "kms:ResourceAliases": "alias/finance-key"
    }
  }
}
```

次の IAM ポリシーステートメントの例では、KMS キーのすべてのエイリアスに「Test」が含まれる場合にのみ、プリンシパルが KMS キーを有効または無効にすることを許可します。このポリシーステートメントは 2 つの条件を使用します。 `ForAllValues` 集合演算子を持つ条件では、KMS キーに関連付けられたすべてのエイリアスに「Test」が含まれている必要があります。 `ForAnyValue` 集合演算子を持つ条件では、KMS キーに 1 つ以上の「Test」を持つエイリアスが含まれている必要があります。 `ForAnyValue` 条件なしの場合、このポリシーステートメントはプリンシパルに、エイリアスがない KMS キーの使用を許可する可能性があります。

```
{
  "Sid": "AliasBasedIAMPolicy",
  "Effect": "Allow",
  "Action": [
    "kms:EnableKey",
```

```
    "kms:DisableKey"
  ],
  "Resource": "arn:aws:kms:*:111122223333:key/*",
  "Condition": {
    "ForAllValues:StringLike": {
      "kms:ResourceAliases": [
        "alias/*Test*"
      ]
    },
    "ForAnyValue:StringLike": {
      "kms:ResourceAliases": [
        "alias/*Test*"
      ]
    }
  }
}
```

kms:ReplicaRegion

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Repli caRegion	文字列 (リスト)	単一値	Replicate Key	キーポリシーと IAM ポリシー

この条件キーを使用して、プリンシパル AWS リージョン が[マルチリージョンキー](#)をレプリケート
できる を制限できます。kms:ReplicaRegion 条件キーは、リクエストの [ReplicaRegion](#) パラメー
タ値に基づいて、[ReplicateKey](#) オペレーションへのアクセスを制御します このパラメータは、AWS
リージョン の新しい[レプリカキー](#)を指定します。

条件の値は、 や などの 1 us-east-1 つ以上の AWS リージョン 名前ap-southeast-2、または
などの名前パターンですeu-*。が AWS KMS サポート AWS リージョン する の名前のリストに
ついては、 の[AWS Key Management Service 「エンドポイントとクォータ」](#)を参照してください
AWS 全般のリファレンス。

例えば、次のキーポリシーステートメントでは kms:ReplicaRegion 条件キーを使用し
て、ReplicaRegion パラメータ値が指定されたリージョンの 1 つである場合にのみ、プリンシパ
ルが [ReplicateKey](#) オペレーションを呼び出すことを許可します。

```
{
```

```
"Effect": "Allow",
"Principal": {
  "AWS": "arn:aws:iam::111122223333:role/Administrator"
},
"Action": "kms:ReplicateKey"
"Resource": "*",
"Condition": {
  "StringEquals": {
    "kms:ReplicaRegion": [
      "us-east-1",
      "eu-west-3",
      "ap-southeast-2"
    ]
  }
}
```

この条件キーは、[ReplicateKey](#) オペレーションへのアクセスのみを制御します。[UpdatePrimaryRegion](#) オペレーションへのアクセスを制御するには、[kms:PrimaryRegion](#) 条件キーを使用します。

kms:RetiringPrincipal

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:RetiringPrincipal	文字列 (リスト)	単一値	CreateGrant	キーポリシーと IAM ポリシー

この条件キーを使用し、リクエストの [RetiringPrincipal](#) のパラメータ値に基づいて、[CreateGrant](#) オペレーションへのアクセスを制御できます。例えば、CreateGrant リクエストの RetiringPrincipal が条件ステートメントの RetiringPrincipal と一致した場合にのみ、KMS キーの使用権限を作成します。

削除プリンシパルを指定するには、AWS プリンシパルの Amazon リソースネーム (ARN) を使用します。有効なプリンシパルには AWS アカウント、IAM ユーザー、IAM ロール、フェデレーティッドユーザー、引き受けたロールユーザーが含まれます。プリンシパル向け ARN 構文については、「IAM ユーザーガイド」の「[IAM の ARN](#)」を参照してください。

次のポリシーステートメント例は、KMS キーに対するグラント (権限付与) の作成をユーザーに許可します。kms:RetiringPrincipal 条件キーが、グラント内の削除プリンシパルが LimitedAdminRole である CreateGrant リクエストへのアクセス許可を制限します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:RetiringPrincipal": "arn:aws:iam::111122223333:role/LimitedAdminRole"
    }
  }
}
```

以下の資料も参照してください。

- [kms:GrantConstraintType](#)
- [kms:GrantIsForAWSResource](#)
- [kms:GrantOperations](#)
- [kms:GranteePrincipal](#)

kms:RotationPeriodInDays

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:RotationPeriodInDays	数値	単一値	EnableKeyRotation	キーポリシーと IAM ポリシー

この条件キーを使用して、プリンシパルが [EnableKeyRotation](#) リクエストの RotationPeriodInDays パラメータに指定できる値を制限できます。

は、各自動キーローテーション日の間の日数RotationPeriodInDaysを指定します。AWS KMS では、90 日から 2560 日までのローテーション期間を指定できますが、kms:RotationPeriodInDays条件キーを使用して、有効範囲内の最小ローテーション期間を強制するなど、ローテーション期間をさらに制限できます。

例えば、次のキーポリシーステートメントでは、kms:RotationPeriodInDays 条件キーの使用により、ローテーション期間が 180 日以内である場合にプリンシパルがキーローテーションを有効化できないようにしています。

```
{
  "Effect": "Deny",
  "Action": "kms:EnableKeyRotation",
  "Principal": "*",
  "Resource": "*",
  "Condition": {
    "NumericLessThanEquals": {
      "kms:RotationPeriodInDays": "180"
    }
  }
}
```

kms:ScheduleKeyDeletionPendingWindowInDays

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:ScheduleKeyDeletionPendingWindowInDays	数値	単一値	ScheduleKeyDeletion	キーポリシーと IAM ポリシー

この条件キーを使用して、プリンシパルが [ScheduleKeyDeletion](#) リクエストの PendingWindowInDays パラメータに指定できる値を制限できます。

は、キーを削除する前に AWS KMS が待機する日数PendingWindowInDaysを指定します。AWS KMS では、7～30 日間の待機期間を指定できますが、kms:ScheduleKeyDeletionPendingWindowInDays条件キーを使用して、有効な範囲内で最小待機期間を適用するなど、待機期間をさらに制限できます。

例えば、次のキーポリシーステートメントで

は、`kms:ScheduleKeyDeletionPendingWindowInDays` 条件キーを使用して、待機期間が 21 日以内の場合にプリンシパルがキーの削除をスケジュールできないようにしています。

```
{
  "Effect": "Deny",
  "Action": "kms:ScheduleKeyDeletion",
  "Principal": "*",
  "Resource": "*",
  "Condition": {
    "NumericLessThanEquals": {
      "kms:ScheduleKeyDeletionPendingWindowInDays": "21"
    }
  }
}
```

kms:SigningAlgorithm

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:SigningAlgorithm	String	単一値	Sign Verify	キーポリシーと IAM ポリシー

`kms:SigningAlgorithm` 条件キーを使用し、リクエストの [SigningAlgorithm](#) パラメータ値に基づいて、[Sign](#) および [Verify](#) オペレーションへのアクセスを制御できます。この条件キーは、非対称 KMS キーペアの外部でパブリックキーを使用して署名を検証するなど AWS KMS、の外部で実行されるオペレーションには影響しません AWS KMS。

次のキーポリシーの例では、リクエストに使用される署名アルゴリズムが RSASSA_PSS アルゴリズム (RSASSA_PSS_SHA512 など) である場合にのみ、`testers` ロールを引き受けることができるユーザーが KMS キーを使用してメッセージに署名できるようにします。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/testers"
  },
}
```

```

{
  "Action": "kms:Sign",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "kms:SigningAlgorithm": "RSASSA_PSS*"
    }
  }
}

```

以下の資料も参照してください。

- [kms:EncryptionAlgorithm](#)
- [the section called “kms:MacAlgorithm”](#)
- [the section called “kms:MessageType”](#)

kms:ValidTo

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:ValidTo	Timestamp	単一値	ImportKey Material	キーポリシーと IAM ポリシー

kms:ValidTo 条件キーは、リクエストの [ValidTo](#) パラメータの値に基づき、[ImportKeyMaterial](#) オペレーションへのアクセスを制御します。これにより、インポートされたキーマテリアルが失効するタイミングが決まります。この値は、[Unix 時間](#)で表現されます。

デフォルトでは、ValidTo パラメータは ImportKeyMaterial リクエストで必要です。ただし、[ExpirationModel](#) パラメータの値が `KEY_MATERIAL_DOES_NOT_EXPIRE`、ValidTo パラメータは無効です。また、[kms:ExpirationModel](#) 条件キーを使用して、ExpirationModel パラメータまたは特定のパラメータの値を要求することもできます。

次のポリシーステートメントの例では、キーマテリアルの KMS キーへのインポートをユーザーに許可します。kms:ValidTo 条件キーは、ImportKeyMaterial リクエストへのアクセス権限を制限します。ここで、ValidTo の値は 1546257599.0 (2018 年 12 月 31 日 午後 11:59:59) 以下となります。

```

{

```

```
"Effect": "Allow",
"Principal": {
  "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
},
"Action": "kms:ImportKeyMaterial",
"Resource": "*",
"Condition": {
  "NumericLessThanEquals": {
    "kms:ValidTo": "1546257599.0"
  }
}
}
```

以下の資料も参照してください。

- [kms:ExpirationModel](#)
- [kms:WrappingAlgorithm](#)
- [kms:WrappingKeySpec](#)

kms:ViaService

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:ViaSe rvice	String	単一値	KMS キーリソー スのオペレー ション	キーポリシーと IAM ポリシー

kms:ViaService 条件キーは、KMS キーの使用を指定された からのリクエストに制限します AWS のサービス。この条件キーは、[転送アクセスセッション](#)にのみ適用されます。各 kms:ViaService 条件キーに 1 つ以上のサービスを指定できます。このオペレーションは KMS キーリソースオペレーションである必要があります。つまり、特定の KMS キーに認可されるオペレーションです。KMS キーリソースオペレーションを識別するには、[アクションとリソースの表](#)で、オペレーションの Resources 列の KMS key の値を探します。

例えば、次のキーポリシーステートメントでは kms:ViaService 条件キーを使用して、リクエストが ExampleRole に代わって米国西部 (オレゴン) リージョンの Amazon EC2 または Amazon RDS

から送信された場合にのみ、[カスターマネージドキー](#)を指定されたアクションで使用できるようにします。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:DescribeKey"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": [
        "ec2.us-west-2.amazonaws.com",
        "rds.us-west-2.amazonaws.com"
      ]
    }
  }
}
```

kms:ViaService 条件キーを使用して、特定のサービスからリクエストが送信された場合に KMS キーの使用許可を拒否することもできます。例えば、次のキーポリシーからのポリシーステートメントでは kms:ViaService 条件キーを使用して、ExampleRole の代わりに AWS Lambda からリクエストが送信された場合に、カスターマネージドキーが Encrypt オペレーションに使用されるのを防ぎます。

```
{
  "Effect": "Deny",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt"
  ],
```

```
"Resource": "*",
"Condition": {
  "StringEquals": {
    "kms:ViaService": [
      "lambda.us-west-2.amazonaws.com"
    ]
  }
}
```

Important

kms:ViaService 条件キーを使用する場合は、サービスが AWS アカウントのプリンシパルの代わりにリクエストを行います。これらのプリンシパルは、次のアクセス許可が必要です。

- KMS キーを使用するアクセス許可。プリンシパルの代わりにサービスがカスターマネージドキーを使用できるようにするために、プリンシパルは統合されたサービスにこれらのアクセス許可を付与する必要があります。詳細については、「[AWS サービスでの AWS KMS 暗号化の使用](#)」を参照してください。
- 統合されたサービスを使用するアクセス権限。と統合する AWS サービスへのアクセス権をユーザーに付与する方法の詳細については AWS KMS、統合サービスのドキュメントを参照してください。

すべての [AWS マネージドキー](#) は、キーポリシードキュメントで kms:ViaService 条件キーを使用します。この条件では、KMS キーを作成したサービスからリクエストされた場合にのみ、KMS キーの使用を許可します。のキーポリシーを表示するには AWS マネージドキー、[GetKeyPolicy](#) オペレーションを使用します。

kms:ViaService 条件キーは IAM とキーポリシーのステートメントで有効です。指定するサービスは、[AWS KMSと統合](#)されていて、kms:ViaService 条件キーをサポートしている必要があります。

kms:ViaService 条件キーをサポートするサービス

次の表に、と統合 AWS KMS され、カスターマネージドキーでの kms:ViaService 条件キーの使用をサポートする AWS サービスを示します。この表のサービスは、すべてのリージョンで利用できるとは限りません。すべての AWS パーティションで AWS KMS ViaService .amazonaws.com 名のサフィックスを使用します。

 Note

この表のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

サービス名	AWS KMS ViaService 名
Amazon API オペレーション	aiops. <i>AWS_region</i> .amazonaws.com
AWS App Runner	apprunner. <i>AWS_region</i> .amazonaws.com
AWS AppFabric	appfabric. <i>AWS_region</i> .amazonaws.com
Amazon AppFlow	appflow. <i>AWS_region</i> .amazonaws.com
AWS Application Migration Service	mgn. <i>AWS_region</i> .amazonaws.com
Amazon Athena	athena. <i>AWS_region</i> .amazonaws.com
AWS Audit Manager	auditmanager. <i>AWS_region</i> .amazonaws.com
Amazon Aurora	rds. <i>AWS_region</i> .amazonaws.com
AWS Backup	backup. <i>AWS_region</i> .amazonaws.com
AWS Backup ゲートウェイ	backup-gateway. <i>AWS_region</i> .amazonaws.com
Amazon Bedrock モデルコピー	bedrock. <i>AWS_region</i> .amazonaws.com
Amazon Chime SDK	chimevoiceconnector. <i>AWS_region</i> .amazonaws.com
AWS Clean Rooms ML	cleanrooms-ml. <i>AWS_region</i> .amazonaws.com

サービス名	AWS KMS ViaService 名
AWS CodeArtifact	codeartifact. <i>AWS_region</i> .amazonaws.com
Amazon CodeGuru Reviewer	codeguru-reviewer. <i>AWS_region</i> .amazonaws.com
Amazon Comprehend	comprehend. <i>AWS_region</i> .amazonaws.com
Amazon Connect	connect. <i>AWS_region</i> .amazonaws.com
Amazon Connect Customer Profiles	profile. <i>AWS_region</i> .amazonaws.com
Amazon Q in Connect	wisdom. <i>AWS_region</i> .amazonaws.com
AWS Database Migration Service (AWS DMS)	dms. <i>AWS_region</i> .amazonaws.com
AWS DeepRacer	deepracer. <i>AWS_region</i> .amazonaws.com
AWS Directory Service	directoryservice. <i>AWS_region</i> .amazonaws.com
Amazon DocumentDB	docdb-elastic. <i>AWS_region</i> .amazonaws.com
Amazon DynamoDB	dynamodb. <i>AWS_region</i> .amazonaws.com
Amazon EC2 Systems Manager (SSM)	ssm. <i>AWS_region</i> .amazonaws.com
Amazon Elastic Block Store (Amazon EBS)	ec2. <i>AWS_region</i> .amazonaws.com (EBSのみ)
Amazon Elastic Container Registry (Amazon ECR)	ecr. <i>AWS_region</i> .amazonaws.com

サービス名	AWS KMS ViaService 名
Amazon Elastic File System (Amazon EFS)	elasticfilesystem. <i>AWS_region</i> .amazonaws.com
Amazon ElastiCache	条件キーバリューに両方の ViaService 名を含めます。 • elasticache. <i>AWS_region</i> .amazonaws.com • dax.<i>AWS_region</i> .amazonaws.com
AWS Elemental MediaTailor	mediatailor. <i>AWS_region</i> .amazonaws.com
AWS エンティティの解決	entityresolution. <i>AWS_region</i> .amazonaws.com
Amazon EventBridge	events. <i>AWS_region</i> .amazonaws.com
Amazon FinSpace	finspace. <i>AWS_region</i> .amazonaws.com
Amazon Forecast	forecast. <i>AWS_region</i> .amazonaws.com
Amazon FSx	fsx. <i>AWS_region</i> .amazonaws.com
AWS Glue	glue. <i>AWS_region</i> .amazonaws.com
AWS Ground Station	groundstation. <i>AWS_region</i> .amazonaws.com
Amazon GuardDuty	malware-protection. <i>AWS_region</i> .amazonaws.com
AWS HealthLake	healthlake. <i>AWS_region</i> .amazonaws.com

サービス名	AWS KMS ViaService 名
AWS IoT SiteWise	iotsitewise. <i>AWS_region</i> .amazonaws.com
Amazon Kendra	kendra. <i>AWS_region</i> .amazonaws.com
Amazon Keyspaces (Apache Cassandra 向け)	cassandra. <i>AWS_region</i> .amazonaws.com
Amazon Kinesis	kinesis. <i>AWS_region</i> .amazonaws.com
Amazon Data Firehose	firehose. <i>AWS_region</i> .amazonaws.com
Amazon Kinesis Video Streams	kinesisvideo. <i>AWS_region</i> .amazonaws.com
AWS Lambda	lambda. <i>AWS_region</i> .amazonaws.com
Amazon Lex	lex. <i>AWS_region</i> .amazonaws.com
AWS License Manager	license-manager. <i>AWS_region</i> .amazonaws.com
Amazon Location Service	geo. <i>AWS_region</i> .amazonaws.com
Amazon Lookout for Equipment	lookoutequipment. <i>AWS_region</i> .amazonaws.com
Amazon Lookout for Metrics	lookoutmetrics. <i>AWS_region</i> .amazonaws.com
Amazon Lookout for Vision	lookoutvision. <i>AWS_region</i> .amazonaws.com
Amazon Macie	macie. <i>AWS_region</i> .amazonaws.com
AWS Mainframe Modernization	m2. <i>AWS_region</i> .amazonaws.com

サービス名	AWS KMS ViaService 名
AWS Mainframe Modernization アプリケーションテスト	apptest. <i>AWS_region</i> .amazonaws.com
Amazon Managed Blockchain	managedblockchain. <i>AWS_region</i> .amazonaws.com
Amazon Managed Streaming for Apache Kafka (Amazon MSK)	kafka. <i>AWS_region</i> .amazonaws.com
Amazon Managed Workflows for Apache Airflow (MWAA)	airflow. <i>AWS_region</i> .amazonaws.com
Amazon MemoryDB	memorydb. <i>AWS_region</i> .amazonaws.com
Amazon Monitron	monitron. <i>AWS_region</i> .amazonaws.com
Amazon MQ	mq. <i>AWS_region</i> .amazonaws.com
Amazon Neptune	rds. <i>AWS_region</i> .amazonaws.com
Amazon Nimble Studio	nimble. <i>AWS_region</i> .amazonaws.com
AWS HealthOmics	omics. <i>AWS_region</i> .amazonaws.com
Amazon OpenSearch Service	es. <i>AWS_region</i> .amazonaws.com , aoss. <i>AWS_region</i> .amazonaws.com
Amazon OpenSearch カスタムパッケージ	custom-packages. <i>AWS_region</i> .amazonaws.com
AWS Proton	proton. <i>AWS_region</i> .amazonaws.com
Amazon Quantum Ledger Database (Amazon QLDB)	qldb. <i>AWS_region</i> .amazonaws.com
Amazon RDS Performance Insights	rds. <i>AWS_region</i> .amazonaws.com

サービス名	AWS KMS ViaService 名
Amazon Redshift	redshift. <i>AWS_region</i> .amazonaws.com
Amazon Redshift クエリエディタ V2	sqlworkbench. <i>AWS_region</i> .amazonaws.com
Amazon Redshift Serverless	redshift-serverless. <i>AWS_region</i> .amazonaws.com
Amazon Rekognition	rekognition. <i>AWS_region</i> .amazonaws.com
Amazon Relational Database Service (Amazon RDS)	rds. <i>AWS_region</i> .amazonaws.com
Amazon 複製データストア	ards. <i>AWS_region</i> .amazonaws.com
Amazon SageMaker AI	sagemaker. <i>AWS_region</i> .amazonaws.com
AWS Secrets Manager	secretsmanager. <i>AWS_region</i> .amazonaws.com
Amazon Security Lake	securitylake. <i>AWS_region</i> .amazonaws.com
Amazon Simple Email Service (Amazon SES)	ses. <i>AWS_region</i> .amazonaws.com
Amazon Simple Notification Service (Amazon SNS)	sns. <i>AWS_region</i> .amazonaws.com
Amazon Simple Queue Service (Amazon SQS)	sqs. <i>AWS_region</i> .amazonaws.com
Amazon Simple Storage Service (Amazon S3)	s3. <i>AWS_region</i> .amazonaws.com
Amazon S3 Tables	s3tables. <i>AWS_region</i> .amazonaws.com

サービス名	AWS KMS ViaService 名
AWS Snowball Edge	importexport. <i>AWS_region</i> .amazonaws.com
AWS Step Functions	states. <i>AWS_region</i> .amazonaws.com
AWS Storage Gateway	storagegateway. <i>AWS_region</i> .amazonaws.com
AWS Systems Manager Incident Manager	ssm-incidents. <i>AWS_region</i> .amazonaws.com
AWS Systems Manager Incident Manager 連絡先	ssm-contacts. <i>AWS_region</i> .amazonaws.com
Amazon Timestream	timestream. <i>AWS_region</i> .amazonaws.com
Amazon Translate	translate. <i>AWS_region</i> .amazonaws.com
AWS Verified Access	verified-access. <i>AWS_region</i> .amazonaws.com
Amazon WorkMail	workmail. <i>AWS_region</i> .amazonaws.com
Amazon WorkSpaces	workspaces. <i>AWS_region</i> .amazonaws.com
Amazon WorkSpaces Thin Client	thinclient. <i>AWS_region</i> .amazonaws.com
Amazon WorkSpaces Web	workspaces-web. <i>AWS_region</i> .amazonaws.com
AWS X-Ray	xray. <i>AWS_region</i> .amazonaws.com

kms:WrappingAlgorithm

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:WrappingAlgorithm	String	単一値	GetParametersForImport	キーポリシーと IAM ポリシー

この条件キーは、リクエストの [WrappingAlgorithm](#) パラメータ値に基づいて、[GetParametersForImport](#) オペレーションへのアクセスを制御します。この条件を使用して、インポートプロセス時にプリンシパルが特定のアルゴリズムを使用してキーマテリアルを暗号化するよう要求できます。異なるラップアルゴリズムを指定すると必要なパブリックキーとインポートトークンのリクエストが失敗します。

次のキーポリシーステートメントの例では、kms:WrappingAlgorithm 条件キーを使用して、GetParametersForImport オペレーションを呼び出すアクセス許可をサンプルユーザーに付与しますが、RSAES_OAEP_SHA_1 ラップアルゴリズムの使用を阻止します。WrappingAlgorithm リクエストの GetParametersForImport が RSAES_OAEP_SHA_1 の場合、オペレーションは失敗します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:GetParametersForImport",
  "Resource": "*",
  "Condition": {
    "StringNotEquals": {
      "kms:WrappingAlgorithm": "RSAES_OAEP_SHA_1"
    }
  }
}
```

以下の資料も参照してください。

- [kms:ExpirationModel](#)
- [kms:ValidTo](#)

- [kms:WrappingKeySpec](#)

kms:WrappingKeySpec

AWS KMS 条件キー	条件の種類	値の型	API オペレーション	ポリシータイプ
kms:WrappingKeySpec	String	単一値	GetParametersForImport	キーポリシーと IAM ポリシー

この条件キーが、リクエストの [WrappingKeySpec](#) パラメータ値に基づいて、[GetParametersForImport](#) オペレーションへのアクセスを制御します。この条件を使用して、インポートプロセス時にプリンシパルが特定のタイプのパブリックキーを使用するよう要求できます。リクエストで別のキータイプを指定すると、エラーになります。

WrappingKeySpec パラメータ値の有効な値は RSA_2048 のみであるため、ユーザーによるこの値の使用を無効にすることで、GetParametersForImport オペレーションを効率的に無効にすることができます。

次のポリシーステートメントの例では、kms:WrappingAlgorithm 条件キーを使用してリクエストの WrappingKeySpec が RSA_4096 になるようにします。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/ExampleRole"
  },
  "Action": "kms:GetParametersForImport",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:WrappingKeySpec": "RSA_4096"
    }
  }
}
```

以下の資料も参照してください。

- [kms:ExpirationModel](#)
- [kms:ValidTo](#)
- [kms:WrappingAlgorithm](#)

AWS KMSAWS Nitro Enclaves の条件キー

[AWS Nitro Enclaves](#) は、機密性の高いデータを保護および処理するために、[enclaves](#) と呼ばれる独立したコンピューティング環境を作成できる Amazon EC2 機能です。は、AWS Nitro Enclaves をサポートする条件キー AWS KMS を提供します。これらの条件キーは、Nitro Enclave AWS KMS のへのリクエストに対してのみ有効です。

署名付き [アテステーションドキュメント](#) を使用して、エンクレーブから

[Decrypt](#)、[DeriveSharedSecret](#)、[GenerateDataKey](#)、[GenerateDataKeyPair](#)、[GenerateRandom](#) のいずれかの API オペレーションを呼び出すとき、これらの API は、アテステーションドキュメントからのパブリックキーのレスポンス内のプレーンテキストを暗号化し、プレーンテキストの代わりに暗号文を返します。この暗号文は、Enclave のプライベートキーを使用してのみ復号できます。詳細については、「[AWS Nitro Enclaves の暗号化アテステーション](#)」を参照してください。

Note

キーの作成時にキーポリシーを指定しない場合、によって AWS KMS キーポリシー AWS が作成されます。この[デフォルトのキーポリシー](#)は、KMS キーを所有 AWS アカウント するにキーへのフルアクセスを付与し、アカウントが IAM ポリシーを使用してキーへのアクセスを許可できるようにします。このポリシーでは、[Decrypt](#) などのすべてのアクションが許可されます。AWS では、KMS キーポリシー[最小特権のアクセス許可](#)に のプリンシパルを適用することをお勧めします。の [KMS キーポリシーアクションを に変更](#) することで、アクセスを制限することもできます `NotAction:kms:Decrypt`。 `kms:*`

次の条件キーを使用すると、署名付きアテステーションドキュメントの内容に基づいて、これらのオペレーションのアクセス許可を制限できます。オペレーションを許可する前に、はエンクレーブのアテステーションドキュメントをこれらの AWS KMS 条件キーの値 AWS KMS と比較します。

kms:RecipientAttestation:ImageSha384

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Recip ientAttes tation:Im ageSha384	String	単一値	Decrypt DeriveSha redSecret Generated ataKey Generated ataKeyPair GenerateR andom	キーポリシーと IAM ポリシー

kms:RecipientAttestation:ImageSha384 条件キーは、リクエスト内の署名付きアテステーションドキュメントからのイメージダイジェストが条件キーの値と一致する場合に、KMS キーを使用して Decrypt、DeriveSharedSecret、GenerateDataKey、GenerateDataKeyPair、GenerateRandom へのアクセスを制御します。ImageSha384 値は、アテステーションドキュメントの PCR0 に対応します。この条件キーは、リクエストの Recipient パラメータが AWS Nitro エンクレーブの署名付き認証ドキュメントを指定している場合にのみ有効です。

この値は、Nitro エンクレーブの へのリクエスト AWS KMS の [CloudTrail イベント](#)にも含まれます。

たとえば、次のキーポリシーステートメントでは、[Decrypt](#)、[DeriveSharedSecret](#)、[GenerateDataKey](#)、[GenerateDataKeyPair](#)、[GenerateRandom](#) の各オペレーション向けの KMS キーの使用を data-processing ロールに許可します。kms:RecipientAttestation:ImageSha384 条件キーでは、リクエスト内のアテステーションドキュメントのイメージダイジェスト値 (PCR0) が条件内のイメージダイジェスト値と一致する場合にのみ、オペレーションを許可します。この条件キーは、リクエストの Recipient パラメータが AWS Nitro エンクレーブの署名付きアテステーションドキュメントを指定している場合にのみ有効です。

リクエストに AWS Nitro エンクレーブからの有効なアテステーションドキュメントが含まれていない場合、この条件が満たされないため、アクセス許可は拒否されます。

```
{
  "Sid" : "Enable enclave data processing",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:role/data-processing"
  },
  "Action": [
    "kms:Decrypt",
    "kms:DeriveSharedSecret",
    "kms:GenerateDataKey",
    "kms:GenerateDataKeyPair",
    "kms:GenerateRandom"
  ],
  "Resource" : "*",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "kms:RecipientAttestation:ImageSha384":
"9fedcba8abcdef7abcdef6abcdef5abcdef4abcdef3abcdef2abcdef1abcdef1abcdef0abcdef1abcdef2abcdef3a
    }
  }
}
```

kms:RecipientAttestation:PCR<PCR_ID>

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
kms:Recip ientAttes tation:PC R<PCR_ID>	String	単一値	Decrypt DeriveSha redSecret Generated ataKey Generated ataKeyPair	キーポリシーと IAM ポリシー

AWS KMS 条件 キー	条件の種類	値の型	API オペレー ション	ポリシータイプ
			GenerateRandom	

`kms:RecipientAttestation:PCR<PCR_ID>` 条件キーは、リクエスト内の署名付きアテステーションドキュメントからのプラットフォーム設定登録 (PCR) が条件キー内の PCR と一致する場合にのみ、KMS キーを使用して `Decrypt`、`DeriveSharedSecret`、`GenerateDataKey`、`GenerateDataKeyPair`、`GenerateRandom` へのアクセスを制御します。この条件キーは、リクエストの `Recipient` パラメータが AWS Nitro エンクレーブから署名付きアテステーションドキュメントを指定する場合にのみ有効です。

この値は、Nitro エンクレーブ AWS KMS の へのリクエストを表す [CloudTrail イベント](#) にも含まれます。

PCR 値を指定するには、次の形式を使用します。PCR ID を条件キー名に連結します。6 [つのエンクレーブ測定値のいずれかを識別する PCR ID](#) または特定のユースケースに定義したカスタム PCR ID を指定できます。PCR 値は、最大 96 バイトの小文字の 16 進文字列である必要があります。

```
"kms:RecipientAttestation:PCR<PCR_ID>": "<PCR_value>"
```

たとえば、次の条件キーは PCR1 の特定の値を指定します。これは、エンクレーブとブートストラッププロセスに使用されるカーネルのハッシュに対応します。

```
kms:RecipientAttestation:PCR1:
  "0x1abcdef2abcdef3abcdef4abcdef5abcdef6abcdef7abcdef8abcdef9abcdef8abcdef7abcdef6abcdef5abcdef"
```

次のキーポリシーステートメントの例では、`data-processing` ロールに [Decrypt](#) オペレーションでの KMS キーの使用を許可します。

このステートメントの `kms:RecipientAttestation:PCR` 条件キーでは、リクエスト内の署名付きアテステーションドキュメントの PCR1 値が条件の `kms:RecipientAttestation:PCR1` 値と一致した場合にのみ、オペレーションを許可します。StringEqualsIgnoreCase ポリシー演算子を使用して、PCR 値の大文字と小文字を区別しない比較を要求します。

リクエストにアテステーションドキュメントが含まれない場合は、この条件が満たされないため、アクセス許可は拒否されます。

```
{
  "Sid" : "Enable enclave data processing",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:role/data-processing"
  },
  "Action": "kms:Decrypt",
  "Resource" : "*",
  "Condition": {
    "StringEqualsIgnoreCase": {
      "kms:RecipientAttestation:PCR1":
        "0x1de4f2dcf774f6e3b679f62e5f120065b2e408dcea327bd1c9dddaea6664e7af7935581474844767453082c6f15"
    }
  }
}
```

最小特権のアクセス許可

KMS キーは機密情報の保護に用いられるため、最小特権アクセスの原則に従うことをお勧めします。キーポリシーを定義する際は、タスクの実行に必要な最小限のアクセス許可のみを委任してください。追加の IAM ポリシーでアクセス許可をさらに制限する予定がある場合にのみ、KMS キーポリシーに対してすべてのアクション (kms:*) を許可します。IAM ポリシーを使用してアクセス許可を管理する予定がある場合は、どのユーザーが IAM ポリシーを作成して IAM プリンシパルにアタッチできるかを制限し、[ポリシーの変更をモニタリング](#)します。

もしキーポリシーと IAM ポリシーの両方ですべてのアクション (kms:*) を許可すると、プリンシパルはその KMS キーに対して管理アクセスと使用の両方の許可を持つことになります。セキュリティのベストプラクティスとして、このようなアクセス許可は特定のプリンシパルにのみ付与することをお勧めします。これを行うには、キーポリシーでプリンシパルを明示的に指名するか、もしくはどのプリンシパルに IAM ポリシーをアタッチするかを制限します。[条件キー](#)を使用してアクセス許可を制限することもできます。例えば、API コールを行うプリンシパルに条件ルールで指定されたタグがある場合、[aws:PrincipalTag](#) を使用してすべてのアクションを許可できます。

ポリシーステートメントの評価方法については AWS、「IAM ユーザーガイド」の「[ポリシー評価口ジック](#)」を参照してください。ポリシーを作成する前にこのトピックを確認して、アクセス権限を持たないプリンシパルへアクセスできるようにするなど、意図しない影響を及ぼすポリシーの作成を避けるように努めてください。

i Tip

本番環境以外の環境でアプリケーションをテストする場合は、[IAM Access Analyzer](#) を使用して IAM ポリシーに最小特権を適用します。

IAM ロールの代わりに IAM ユーザーを使用する場合は、長期的な認証情報の脆弱性を軽減するために AWS [多要素認証](#) (MFA) を有効にすることを強くお勧めします。MFA の使用により、以下が可能となります。

- キー削除のスケジューリングなどの特権アクションを実行する前に、MFA を使用して認証情報を検証するようユーザーに義務付けることができます。
- 管理者アカウントのパスワードと MFA デバイスの所有権を複数の人の間で分割する「分割認証」を実装することができます。

詳細はこちら

- [AWS ジョブ関数の マネージドポリシー](#)
- [Techniques for writing least privilege IAM policies](#)

最小特権のアクセス許可の実装

KMS キーを使用するアクセス許可を AWS サービスに付与する場合は、そのアクセス許可が、サービスがユーザーに代わってアクセスする必要があるリソースに対してのみ有効であることを確認します。この最小特権戦略は、AWS サービス間でリクエストが渡されたときに KMS キーが不正に使用されるのを防ぐのに役立ちます。

最小特権戦略を実装するには、AWS KMS 暗号化コンテキスト条件キーとグローバルソース ARN またはソースアカウント条件キーを使用することをお勧めします。

暗号化コンテキスト条件キーの使用

リソースを使用する際 AWS KMS に最小特権のアクセス許可を実装する最も効果的な方法は、プリンシパルが AWS KMS 暗号化オペレーションを呼び出すことを許可する [kms:EncryptionContext:context-key](#) または [kms:EncryptionContextKeys](#) 条件キーをポリシーに含めることです。これらの条件キーは、リソースが暗号化されるときに暗号文にバインドされる [暗号化コンテキスト](#) にアクセス許可を関連付けるため、特に効果的です。

暗号化コンテキスト条件キーは、ポリシーステートメントのアクションが [CreateGrant](#) であるか、[GenerateDataKey](#) AWS KMS や [Decrypt](#) などのオペレーションなどの EncryptionContext パラメータを使用する対称暗号化オペレーションの場合にのみ使用します。(サポートされているオペレーションのリストについては、[kms:EncryptionContext:context-key](#) または [kms:EncryptionContextKeys](#) を参照してください) これらの条件キーを使用して他のオペレーション ([DescribeKey](#) など) を許可する場合、アクセス許可は拒否されます。

サービスがリソースを暗号化するとき使用する暗号化コンテキストに値を設定します。この情報は通常、サービスのドキュメントのセキュリティに関する章で確認できます。例えば、[AWS Proton の暗号化コンテキスト](#)は、Proton AWS リソースとそれに関連するテンプレートを識別します。[AWS Secrets Manager 暗号化コンテキスト](#)が、シークレットとそのバージョンを識別します。[Amazon Location の暗号化コンテキスト](#)が、トラッカーやコレクションを識別します。

次のキーポリシーステートメントの例では、承認されたユーザーの代わりに Amazon Location Service がグラントの作成を許可しています。このポリシーステートメントでは、[kms:ViaService](#)、[kms:CallerAccount](#)、および `kms:EncryptionContext:context-key` 条件キーを使用して特定のトラッカーリソースにアクセス許可を結び付けることで、アクセス許可が制限されます。

```
{
  "Sid": "Allow Amazon Location to create grants on behalf of authorized users",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/LocationTeam"
  },
  "Action": "kms:CreateGrant",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:ViaService": "geo.us-west-2.amazonaws.com",
      "kms:CallerAccount": "111122223333",
      "kms:EncryptionContext:aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:tracker/SAMPLE-Tracker"
    }
  }
}
```

aws:SourceArn または aws:SourceAccount 条件キーの使用

キーポリシーステートメントのプリンシパルが [AWS のサービスプリンシパル](#) になる場合は、`kms:EncryptionContext:context-key` 条件キーに加えて [aws:SourceArn](#) または

[aws:SourceAccount](#) グローバル条件キーの使用を強くお勧めします。ARN とアカウント値は、リクエストが別の AWS サービス AWS KMS から送信された場合にのみ認可コンテキストに含まれます。この条件の組み合わせにより、最小特権のアクセス許可が実装され、可能性のある「[混乱した代理](#)」シナリオが回避されます。サービスプリンシパルは通常、キーポリシーでプリンシパルとして使用されませんが、それ AWS CloudTrailを必要とする AWS サービスもあります。

`aws:SourceArn` または `aws:SourceAccount` グローバル条件キーを使用する場合、暗号化されているリソースの Amazon リソースネーム (ARN) またはアカウントに値を設定します。例えば、証跡を暗号化するための AWS CloudTrail アクセス許可を付与するキーポリシーステートメントでは、その証跡の ARN に `aws:SourceArn` の値を設定します。可能な限り、より具体的な `aws:SourceArn` を使用してください。ARN またはワイルドカード文字を使用した ARN パターンに値を設定します。リソースの ARN が不明の場合は、代わりに `aws:SourceAccount` を使用してください。

Note

リソース ARN に AWS KMS キーポリシーで許可されていない文字が含まれている場合、そのリソース ARN を `aws:SourceArn` 条件キーの値で使用することはできません。その代わりに `aws:SourceAccount` 条件キーを使用してください。キーポリシードキュメントのルールに関する詳細については、「[キーポリシー形式](#)」を参照してください。

次のキーポリシーの例では、アクセス許可を取得するプリンシパルは AWS CloudTrail サービスプリンシパル、`cloudtrail.amazonaws.com` になります。最小特権を実装するために、このポリシーでは `aws:SourceArn` および `kms:EncryptionContext:context-key` 条件キーが使用されます。ポリシーステートメントにより、CloudTrail が KMS キーを使用して、証跡の暗号化に利用する[データキーを生成](#)できるようになります。`aws:SourceArn` および `kms:EncryptionContext:context-key` の条件は個別に評価されます。指定されたオペレーションの KMS キーを使用するリクエストでは、両方の条件が満たされている必要があります。

サンプルアカウント (111122223333) および `us-west-2` リージョンの `finance` 証跡に対するサービスのアクセス許可を制限する場合、このポリシーステートメントで `aws:SourceArn` 条件キーを特定の証跡の ARN に設定します。条件ステートメントでは、ARN 内のすべての要素が一致するときに個別に評価されるように [ArnEquals](#) 演算子を使用されます。この例では、特定のアカウントおよびリージョンの証跡に対するアクセス許可を制限するために、`kms:EncryptionContext:context-key` 条件キーも使用されています。

このキーポリシーを使用する前に、サンプルアカウント ID、リージョン、および証跡名をアカウントの有効な値に置き換えてください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCloudTrailToEncryptLogs",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudtrail.amazonaws.com"
      },
      "Action": "kms:GenerateDataKey",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": [
            "arn:aws:cloudtrail:us-west-2:111122223333:trail/finance"
          ]
        },
        "StringLike": {
          "kms:EncryptionContext:aws:cloudtrail:arn": [
            "arn:aws:cloudtrail:*:111122223333:trail/*"
          ]
        }
      }
    }
  ]
}
```

の ABAC AWS KMS

属性ベースのアクセスコントロール (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。は、KMS キーに関連付けられたタグとエイリアスに基づいてカスターマネージドキーへのアクセスを制御できるようにすることで、ABAC AWS KMS をサポートします。で ABAC を有効にするタグとエイリアスの条件キーは、ポリシーを編集したり許可を管理したりすることなく、プリンシパルに KMS キーの使用を許可する強力な柔軟な方法 AWS KMS を提供します。ただし、プリンシパ

ルが誤ってアクセスを許可されたり拒否されないように、注意してこれらの機能を使用する必要があります。

ABAC を使用する場合は、タグとエイリアスを管理する許可が、アクセス制御許可になることに注意してください。タグまたはエイリアスに依存するポリシーをデプロイする前に、すべての KMS キーの既存のタグとエイリアスを把握していることを確認してください。エイリアスの追加、削除、更新時、およびキーのタグ付けおよびタグ解除時には、妥当な予防措置を講じます。タグとエイリアスを必要とするプリンシパルにのみ管理する許可を付与し、管理できるタグとエイリアスを制限します。

メモ

ABAC を に使用するときには AWS KMS、タグとエイリアスを管理するアクセス許可をプリンシパルに付与することに注意してください。タグまたはエイリアスを変更すると、KMS キーに対するアクセス許可を、許可または拒否する可能性があります。キーポリシーを変更したり、グラントを作成したりする許可を持たないキー管理者も、タグやエイリアスを管理する許可があれば、KMS キーへのアクセスを制御できます。

タグとエイリアスの変更が KMS キーの認可に影響を及ぼすまでに最長 5 分かかることがあります。最近の変更は、認可に影響を与える前に API オペレーションで表示される場合があります。

エイリアスに基づいて KMS キーへのアクセスを制御するには、条件キーを使用する必要があります。ポリシーステートメントの Resource 要素のエイリアスを使用して KMS キーを表すことはできません。エイリアスが Resource 要素で表示される場合、ポリシーステートメントは、関連付けられた KMS キーではなく、エイリアスに適用されます。

詳細はこちら

- 例を含む ABAC AWS KMS のサポートの詳細については、[エイリアスを使用して KMS キーへのアクセスを制御する](#)「」および「」を参照してください[タグを使用して KMS キーへのアクセスを制御する](#)。
- タグを使用してリソースへのアクセス AWS を制御する方法の詳細については、IAM ユーザーガイドの「[ABAC for AWSとは](#)」および「[リソースタグを使用した AWS リソースへのアクセスの制御](#)」を参照してください。

の ABAC 条件キー AWS KMS

タグとエイリアスに基づいて KMS キーへのアクセスを認可するには、キーポリシーまたは IAM ポリシーで次の条件キーを使用します。

ABAC 条件キー	説明	ポリシータイプ	AWS KMS オペレーション
aws:ResourceTag	KMS キーのタグ (キーと値) が、ポリシーのタグ (キーと値) またはタグパターンと一致する	IAM ポリシーのみ	KMS キーリソースのオペレーション ²
aws:RequestTag/tag-key	リクエスト内のタグ (キーと値) が、ポリシー内のタグ (キーと値) またはタグパターンと一致する	キーポリシーと IAM ポリシー ¹	TagResource 、 UntagResource
aws:TagKeys	リクエスト内のタグキーが、ポリシーのタグキーと一致する	キーポリシーと IAM ポリシー ¹	TagResource 、 UntagResource
kms:ResourceAliases	KMS キーに関連付けられたエイリアスが、ポリシーのエイリアスまたはエイリアスパターンと一致する	IAM ポリシーのみ	KMS キーリソースのオペレーション ²
kms:RequestAlias	リクエスト内の KMS キーを表すエイリアスが、ポリシーのエイリアスまたはエイリアスパターンと一致する。	キーポリシーと IAM ポリシー ¹	Cryptographic operation 、 DescribeKey 、 GetPublicKey

¹ キーポリシーで使用できる条件キーは、IAM ポリシーでも使用できます。ただし、[キーポリシーによって許可される場合](#)に限ります。

² KMS キーのリソースオペレーションは、特定の KMS キーに対して認可されるオペレーションです。KMS キーリソースオペレーションを識別するには、[AWS KMS アクセス許可の表](#)で、オペレーションの Resources 列の KMS キー値を探します。

例えば、これらの条件キーを使用して、次のポリシーを作成できます。

- 特定のエイリアスまたはエイリアスパターンを持つ KMS キーを使用するアクセス許可を付与する `kms:ResourceAliases` を備えた IAM ポリシー。これは、タグに依存するポリシーとは少し異なります。ポリシーでエイリアスパターンを使用できますが、各エイリアスは AWS アカウントおよびリージョンで一意である必要があります。これにより、KMS キーのキー ARN をポリシーステートメントに表示せずに、選択した KMS キーのセットにポリシーを適用できます。セットの KMS キーを追加または削除するには、KMS キーのエイリアスを変更します。
- Encrypt リクエストがエイリアスを使用して KMS キーを識別する場合にのみ、Encrypt オペレーションでプリンシパルの KMS キー使用を許可する `kms:RequestAlias` を備えたキーポリシー。
- 特定のタグキーとタグ値を持つ KMS キーを使用するアクセス許可を拒否する `aws:ResourceTag/tag-key` を備えた IAM ポリシー。これにより、KMS キーのキー ARN をポリシーステートメントに表示せずに、選択した KMS キーのセットにポリシーを適用できます。セットの KMS キーを追加または削除するには、KMS キーをタグ付けまたはタグ解除します。
- プリンシパルが "Purpose"="Test" KMS キータグのみを削除することを許可する `aws:RequestTag/tag-key` を備えた IAM ポリシー。
- Restricted タグキーで KMS キーをタグ付けまたはタグ解除する許可を拒否する `aws:TagKeys` を備えた IAM ポリシー。

ABAC は、アクセス管理を柔軟かつスケーラブルにします。例えば、`aws:ResourceTag/tag-key` 条件キーを使用して、KMS キーが `Purpose=Test` タグを持つ場合にのみ、プリンシパルが指定されたオペレーションに KMS キーを使用することを許可する IAM ポリシーを作成します。このポリシーは、AWS アカウントのすべてのリージョンの KMS キーに適用されます。

ユーザーまたはロールにアタッチされると、以下の IAM ポリシーにより、プリンシパルは指定されたオペレーションの `Purpose=Test` タグで既存の KMS キーを使用できるようになります。新規または既存の KMS キーにこのアクセスを提供するためにポリシーを変更する必要はありません。単に、KMS キーに `Purpose=Test` タグをアタッチします。同様に、`Purpose=Test` タグでこのアクセスを KMS キーから削除するには、タグを編集または削除します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Purpose": "Test"
        }
      }
    }
  ]
}
```

ただし、この機能を使用する場合は、タグとエイリアスを管理する際に注意が必要です。タグまたはエイリアスを追加、変更、削除する際に、KMS キーへのアクセスを誤って許可または拒否する可能性があります。キーポリシーを変更したり、グラントを作成したりする許可を持たないキー管理者も、タグとエイリアスを管理する許可があれば、KMS キーへのアクセスを制御できます。このリスクを軽減するには、[タグおよびエイリアスを管理する許可の制限](#)を検討します。例えば、選択したプリンシパルのみが Purpose=Test タグを管理できるようにします。詳細については、「[エイリアスを使用して KMS キーへのアクセスを制御する](#)」および「[タグを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

タグまたはエイリアス

AWS KMS は、タグとエイリアスを含む ABAC をサポートしています。どちらのオプションも、柔軟でスケーラブルなアクセス制御方法を提供しますが、互いに若干異なります。

タグを使用するか、特定の使用パターンに基づいてエイリアス AWS を使用するかを選択できます。例えば、すでにほとんどの管理者にタグ付け許可を付与している場合は、エイリアスに基づいて認可

の方法を制御する方が簡単です。または、[KMS キーごとのエイリアス](#)のクォータに近づいている場合、タグに基づく認可の方法の方が得策かもしれません。

以下は、一般的な利点です。

タグベースのアクセスコントロールの利点

- 異なるタイプの AWS リソースに対して同じ認可メカニズム。

同じタグまたはタグキーを使用して、Amazon Relational Database Service (Amazon RDS) クラスター、Amazon Elastic Block Store (Amazon EBS) ボリューム、KMS キーなど、複数のリソースタイプへのアクセスを制御できます。この機能により、従来のロールベースのアクセス制御よりも柔軟性が高い、複数の異なる認可モデルが可能になります。

- KMS キーグループへのアクセスを認可します。

タグを使用すると、同じ AWS アカウント およびリージョンの KMS キーグループへのアクセスを管理できます。選択した KMS キーに同じタグまたはタグキーを割り当てます。次に、タグまたはタグキーに基づいて、シンプルで保守が容易なポリシーステートメントを作成します。認可グループの KMS キーを追加または削除するには、タグを追加または削除します。ポリシーを編集する必要はありません。

エイリアスベースのアクセス制御の利点

- エイリアスに基づいて暗号化オペレーションへのアクセスを認可します。

属性に対するほとんどのリクエストベースのポリシー条件 ([aws:RequestTag/tag-key](#) など) は、属性を追加、編集、削除するオペレーションにのみ影響します。ただし、[kms:RequestAlias](#) 条件キーは、リクエストの KMS キーの識別に使用されるエイリアスに基づき、暗号化オペレーションへのアクセスを制御します。例えば、Encrypt オペレーションで KMS キーを使用するプリンシパルに、KeyId パラメータ値が `alias/restricted-key-1` の場合にのみ、アクセス許可を付与できます。この条件を満たすには、次のすべてが必要です。

- KMS キーがそのエイリアスに関連付けられている必要があります。
- リクエストで、KMS キーを識別するためにエイリアスを使用する必要があります。
- `kms:RequestAlias` を条件として、プリンシパルが KMS キーを使用するアクセス許可が必要です。

これは、アプリケーションが一般的にエイリアス名またはエイリアス ARN を使用して KMS キーを参照する場合に特に便利です。

- きわめて限定されたアクセス許可を付与します。

エイリアスは、AWS アカウント および リージョンで一意である必要があります。結果として、プリンシパルに、エイリアスに基づいて KMS キーへのアクセスを許可することは、タグに基づいてプリンシパルにアクセスを許可するよりもはるかに制限が厳しくなります。エイリアスとは異なり、タグは同じアカウントとリージョン内の複数の KMS キーに割り当てることができます。選択すると、エイリアスパターン (alias/test* など) を使用して、プリンシパルに同じアカウントとリージョン内の KMS キーグループへのアクセスを許可します。ただし、特定のエイリアスへのアクセスを許可または拒否すると、KMS キーのきわめて厳密な制御が可能になります。

の ABAC のトラブルシューティング AWS KMS

タグとエイリアスに基づいて KMS キーへのアクセスを制御することは、便利で強力です。ただし、いくつかの予測可能なエラーが発生しやすいため、予防する必要があります。

タグの変更によりアクセスが変更される

タグが削除されるかその値が変更されると、そのタグのみに基づいて KMS キーにアクセスするプリンシパルは、KMS キーへのアクセスを拒否されます。これは、拒否ポリシーステートメントに含まれるタグが KMS キーに追加された場合にも発生します。ポリシー関連のタグを KMS キーに追加すると、KMS キーへのアクセスを拒否されるプリンシパルにアクセスを許可できます。

例えば、プリンシパルに、Project=Alpha タグに基づく KMS キーへのアクセス許可 (以下の例の、IAM ポリシーステートメントによって付与されるアクセス許可など) があるとします。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:ap-southeast-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
```

```
        "aws:ResourceTag/Project": "Alpha"
      }
    }
  }
}
```

タグがその KMS キーから削除されるか、タグの値が変更されると、プリンシパルには指定されたオペレーションで KMS キーを使用するアクセス許可がなくなります。これは、プリンシパルがカスターマネージドキーを使用する AWS サービスでデータの読み取りまたは書き込みを試みると明らかになる場合があります。タグの変更を追跡するには、CloudTrail ログで [TagResource](#) または [UntagResource](#) [エントリ](#)を確認します。

ポリシーを更新せずにアクセスを復元するには、KMS キーのタグを変更します。このアクションは、AWS KMS 全体で有効な期間中、短い期間を除いて影響は最小限です。このようなエラーを防ぐには、タグ付けとタグ解除の許可を必要なプリンシパルのみに付与し、[タグ付け許可を、管理する必要があるタグに制限します](#)。タグを変更する前にポリシーを検索して、タグに依存するアクセスを検出し、タグを持つすべてのリージョンで KMS キーを取得します。特定のタグが変更されるときは、Amazon CloudWatch アラームの作成を検討してください。

エイリアスの変更によるアクセス変更

エイリアスが削除されたり、別の KMS キーに関連付けられている場合、そのエイリアスのみに基づいて KMS キーにアクセスするプリンシパルは、KMS キーへのアクセスを拒否されます。これは、KMS キーに関連付けられたエイリアスが拒否ポリシーステートメントに含まれる場合にも発生します。ポリシー関連のエイリアスを KMS キーに追加すると、KMS キーへのアクセスを拒否されるプリンシパルへのアクセスを許可できます。

例えば、次の IAM ポリシーステートメントは、[kms:ResourceAliases](#) 条件キーを使用して、指定されたエイリアスを持つアカウントの異なるリージョンの KMS キーへのアクセスを許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
```



```
"Action": [
  "kms:List*",
  "kms:Describe*",
  "kms:Decrypt"
],
"Resource": "arn:aws:kms:*:111122223333:key/*",
"Condition": {
  "ForAnyValue:StringEquals": {
    "kms:ResourceAliases": [
      "alias/ProjectAlpha",
      "alias/ProjectAlpha_Test",
      "alias/ProjectAlpha_Dev"
    ]
  }
}
```

エイリアスの変更を追跡するには、CloudTrail ログで、[CreateAlias](#)、[UpdateAlias](#)、[DeleteAlias](#) エントリを確認します。

ポリシーを更新せずにアクセスを復元するには、KMS キーに関連付けられたエイリアスを変更します。各エイリアスは、アカウントおよびリージョン内の 1 つの KMS キーのみにしか関連付けできないため、エイリアスの管理は、タグの管理よりも若干難しくなります。ある KMS キーの一部のプリンシパルへのアクセスを復元すると、異なる KMS キーの同じプリンシパルまたは他のプリンシパルへのアクセスが拒否されることがあります。

このエラーを防ぐには、エイリアス管理許可を必要なプリンシパルのみに付与し、[エイリアス管理許可の制限](#)を管理する必要があるエイリアスに制限します。エイリアスを更新または削除する前に、ポリシーを検索してエイリアスに依存するアクセスを検出し、エイリアスに関連付けられているすべてのリージョンで KMS キーを検索します。

エイリアスクォータによりアクセスが拒否された

[kms:ResourceAliases](#) 条件によって KMS キーの使用を認可されているユーザーには、KMS キーがそのアカウントまたはリージョンの、デフォルトの[KMS キーあたりのエイリアスクォータ](#)を超えた場合、AccessDenied の例外が適用されます。

アクセスを復元するには、KMS キーに関連付けられたエイリアスを削除して、クォータに適合させます。または、別のメカニズムを使用して、ユーザーに KMS キーへのアクセスを許可します。

認可変更の遅延

タグとエイリアスを変更すると、KMS キーの認可に影響を及ぼすまでに最大 5 分かかる場合があります。結果として、タグやエイリアスの変更が認可に影響を与える前に、API オペレーションからのレスポンスに反映される可能性があります。この遅延は、ほとんどの AWS KMS オペレーションに影響する短い結果整合性遅延よりも長くなる可能性があります。

例えば、特定のプリンシパルに "Purpose"="Test" タグで KMS キーの使用を許可する IAM ポリシーなどです。次に、"Purpose"="Test" タグを KMS キーに追加します。[TagResource](#) オペレーションが完了し、[ListResourceTags](#) レスポンスによって、タグが KMS キーに割り当てられていることが確認されても、プリンシパルは KMS キーに最大 5 分間アクセスできない場合があります。

エラーを防ぐには、この予想される遅延をコードに組み込みます。

エイリアスの更新によるリクエストの失敗

エイリアスを更新するときは、既存のエイリアスを別の KMS キーに関連付けます。

[エイリアス名](#)または[エイリアス ARN](#)を指定する [Decrypt](#) および [ReEncrypt](#) リクエストは、エイリアスが暗号文を暗号化していない KMS キーに関連付けられたため、失敗することがあります。この状況は通常、`IncorrectKeyException` または `NotFoundException` を返します。または、リクエストに `KeyId` または `DestinationKeyId` パラメータがない場合、発信者が暗号文を暗号化した KMS キーにアクセスできなくなったため、`AccessDenied` の例外により、オペレーションは失敗する可能性があります。

CloudTrail ログで [CreateAlias](#)、[UpdateAlias](#)、[DeleteAlias](#) ログエントリを探して、変更を追跡できます。[ListAliases](#) レスポンスの `LastUpdatedDate` フィールド値を使用して、変更を検出することもできます。

例えば、次の [ListAliases](#) レスポンスの例では、`kms:ResourceAliases` 条件の `ProjectAlpha_Test` エイリアスが更新されたことを確認できます。この結果、エイリアスに基づくアクセス許可を持つプリンシパルは、以前に関連付けられた KMS キーにアクセスできなくなります。代わりに、新しく関連付けられた KMS キーにアクセスできます。

```
$ aws kms list-aliases --query 'Aliases[?starts_with(AliasName, `alias/ProjectAlpha`)]'

{
  "Aliases": [
    {
```

```

        "AliasName": "alias/ProjectAlpha_Test",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ProjectAlpha_Test",
        "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": 1566518783.394,
        "LastUpdatedDate": 1605308931.903
    },
    {
        "AliasName": "alias/ProjectAlpha_Restricted",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ProjectAlpha_Restricted",
        "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": 1553410800.010,
        "LastUpdatedDate": 1553410800.010
    }
]
}

```

この変更を回避するのは簡単ではありません。エイリアスを再度更新して、元の KMS キーに関連付けることができます。ただし、オペレーションを行う前に、その変更が現在関連付けられている KMS キーに及ぼす影響を考慮する必要があります。プリンシパルが暗号化オペレーションで後者の KMS キーを使用した場合は、引き続きそのキーにアクセスする必要がある可能性もあります。この場合はポリシーを更新し、プリンシパルに両方の KMS キーを使用するアクセス許可があることを確認します。

エイリアスを更新する前にポリシーを検索して、エイリアスに依存するアクセスを検出することで、このようなエラーを防ぐことができます。次に、エイリアスに関連付けられているすべてのリージョンで KMS キーを取得します。エイリアス管理許可を、必要とするプリンシパルにのみ付与し、[エイリアス管理許可の制限](#)を、管理する必要があるエイリアスに設定します。

の RBAC AWS KMS

ロールベースのアクセスコントロール (RBAC) とは、それぞれのユーザーに対して業務の実行に必要な最小限なアクセス許可のみを提供する認証戦略です。AWS KMS は、キー用途に基づくきめ細かなアクセス許可を[キーポリシー](#)内で指定するキーアクセス制御を可能にすることで、RBAC をサポートします。キーポリシーは、キーへのアクセスを許可するリソース、アクション、効果、プリンシパル、およびオプション条件を指定します。

RBAC を実装するには AWS KMS、キーユーザーとキー管理者のアクセス許可を分離することをお勧めします。

Key users

例えば、以下のキーポリシーステートメントは、ExampleUserRole IAM ロールに KMS キーの使用を許可します。

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleUserRole"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

キーユーザーが必要とするアクセス許可は、恐らくこの例のユーザーよりも少ないでしょう。そのユーザーが必要とするアクセス許可のみを割り当ててください。アクセス許可をさらに絞り込むには、次の質問を使用します。

- どの IAM プリンシパル (ロールまたはユーザー) がそのキーにアクセスする必要がありますか？
- 各プリンシパルがそのキーを用いて実行する必要があるアクションはどんなものがありますか？ 例えば、そのプリンシパルに必要なのは暗号化と署名のアクセス許可だけですか？
- ユーザーは人間または AWS のサービスですか？ サービスの場合は AWS、[条件キー](#)を使用して、キーの使用を特定の AWS サービスに制限できます。

Key administrators

以下のキーポリシー例は、ExampleAdminRole IAM ロールが KMS キーを管理することを許可します。

```
{
    "Sid": "Allow access for Key Administrators",
    "Effect": "Allow",
```

```
"Principal": {
  "AWS": "arn:aws: iam::111122223333:role/ExampleAdminRole"
},
"Action": [
  "kms:Create*",
  "kms:Describe*",
  "kms:Enable*",
  "kms:List*",
  "kms:Put*",
  "kms:Update*",
  "kms:Revoke*",
  "kms:Disable*",
  "kms:Get*",
  "kms>Delete*",
  "kms:TagResource",
  "kms:UntagResource",
  "kms:ScheduleKeyDeletion",
  "kms:CancelKeyDeletion"
],
"Resource": "*"
}
```

キー管理者に必要なアクセス許可は、恐らくこの例の管理者より少ないでしょう。そのキー管理者が必要とするアクセス許可のみを割り当ててください。

各ユーザーが自らのロールの履行に必要なアクセス許可のみを付与してください。ユーザーのアクセス許可は、キーがテスト環境で使用されるか本番環境で使用されるかによっても異なります。特定の非本番環境で制限の少ないアクセス許可を使用する場合は、そのポリシーを本番環境にリリースする前にテストするプロセスを実装します。

詳細

- [IAM ID \(ユーザー、ユーザーグループ、ロール\)](#)
- [アクセスコントロールのタイプ](#)

他のアカウントのユーザーに KMS キーの使用を許可する

別の のユーザーまたはロール AWS アカウント に、 アカウントの KMS キーの使用を許可できます。クロスアカウントアクセスには、KMS キーのキーポリシーと、外部ユーザーアカウントの IAM ポリシーのアクセス許可が必要です。

クロスアカウント許可は、以下のオペレーションに対してのみ有効です。

- [暗号化オペレーション](#)
- [CreateGrant](#)
- [DescribeKey](#)
- [GetKeyRotationStatus](#)
- [GetPublicKey](#)
- [ListGrants](#)
- [RetireGrant](#)
- [RevokeGrant](#)

別のアカウントのユーザーに他のオペレーションのアクセス許可を付与しても、それらのアクセス許可には効果がありません。例えば、別のアカウントのプリンシパルに、IAM ポリシーの [kms:ListKeys](#) アクセス許可を付与した場合、またはキーポリシーの KMS キーで [kms:ScheduleKeyDeletion](#) を付与した場合、リソースでこれらのオペレーションを呼び出そうとするユーザーの試行は失敗します。

AWS KMS オペレーションに異なるアカウントで KMS キーを使用する方法の詳細については、[AWS KMS アクセス許可](#)「」および「」の「クロスアカウント使用」列を参照してください[他のアカウントで KMS キーを使用する](#)。また、[AWS Key Management Service API リファレンス](#)の各 API の説明に、クロスアカウント使用のセクションがあります。

 Warning

KMS キー使用のアクセス許可をプリンシパルに付与する際は注意してください。可能な限り、最小権限の原則に従ってください。オペレーションに必要な KMS キーのみに、アクセスを許可します。

また、使い慣れていない KMS キー、特に別のアカウントの KMS キーを使用する際にも注意してください。悪意のあるユーザーは、ユーザーまたはユーザーのアカウントに関する情報を取得するために、KMS キーの使用許可を付与する可能性があります。

ポリシーを使用してアカウント内のリソースを保護する方法については、「[IAM ポリシーのベストプラクティス](#)」を参照してください。

別のアカウントのユーザーとロールに KMS キーを使用するアクセス許可を付与するには、2 つの異なるタイプのポリシーを使用する必要があります。

- KMS キーのキーポリシーでは、KMS キーを使用するアクセス許可を、外部アカウント (または外部アカウントのユーザーとロール) に付与する必要があります。キーポリシーは、KMS キーを所有するアカウントにあります。
- 外部アカウントの IAM ポリシーは、キーポリシーのアクセス権限をそのユーザーとロールに委任する必要があります。これらのポリシーは外部アカウントで設定され、そのアカウントのユーザーとロールにアクセス許可を与えます。

キーポリシーによって、KMS キーにアクセスできるユーザーが決定されます。IAM ポリシーによって、KMS キーにアクセスするユーザーが決定されます。キーポリシーも IAM ポリシーも十分ではありません。両方を変更する必要があります。

キーポリシーを編集するために、AWS Management Console の[ポリシービュー](#)、[CreateKey](#)、[PutKeyPolicy](#) オペレーションを使用できます。

IAM ポリシーの編集については、「[での IAM ポリシーの使用 AWS KMS](#)」を参照してください。

キーポリシーと IAM ポリシーが連携して、別のアカウントで KMS キーの使用を許可する方法を示す例については、[例 2: ユーザーが別の で KMS キーを使用するアクセス許可を持つロールを引き受ける AWS アカウント](#) を参照してください。

結果のクロスアカウント AWS KMS オペレーションは、[AWS CloudTrail ログ](#)の KMS キーで確認できます。他のアカウントで KMS キーを使用するオペレーションは、発信者のアカウントと KMS キー所有者のアカウントの両方に記録されます。

トピック

- [ステップ 1: ローカルアカウントにキーポリシーステートメントを追加する](#)
- [ステップ 2: 外部アカウントに IAM ポリシーを追加する](#)
- [外部 KMS キーの使用を許可する AWS のサービス](#)
- [他のアカウントで KMS キーを使用する](#)

Note

このトピックの例は、キーポリシーと IAM ポリシーを併用して、KMS キーへのアクセス権を付与し、それらを制限する方法を示しています。これらの一般的な例は、特定の が KMS キーに対して AWS のサービス 必要とするアクセス許可を表すことを目的としたものではありません。

りません。に必要なアクセス許可の詳細については AWS のサービス、サービスドキュメントの暗号化トピックを参照してください。

ステップ 1: ローカルアカウントにキーポリシーステートメントを追加する

KMS キーのキーポリシーは、KMS キーにアクセスできるユーザーと、実行できるオペレーションの主要な決定要因です。キーポリシーは常に、KMS キーを所有するアカウントにあります。IAM ポリシーとは異なり、キーポリシーはリソースを指定しません。リソースは、キーポリシーに関連付けられている KMS キーです。クロスアカウント許可を付与する場合、KMS キーのキーポリシーでは、KMS キーを使用するアクセス許可を、外部アカウント (または外部アカウントのユーザーとロール) に付与する必要があります。

KMS キーを使用する許可を外部アカウントに付与するには、外部アカウントを指定するステートメントをキーポリシーに追加します。キーポリシーの Principal 要素に、外部アカウントの Amazon リソースネーム (ARN) を入力します。

キーポリシーで外部アカウントを指定すると、外部アカウントの IAM 管理者は IAM ポリシーを使用して、外部アカウントのすべてのユーザーおよびロールにこれらのアクセス権限を委任できます。また、ユーザーおよびロールが実行できるキーポリシーで指定されたアクションを決定することもできます。

外部アカウントとそのプリンシパルに付与されたアクセス許可は、KMS キーとそのキーポリシーをホストするリージョンで外部アカウントが有効になっている場合にのみ有効です。デフォルトで有効になっていないリージョン (「オプトインリージョン」) については、「AWS 全般のリファレンス」の「[AWS リージョンの管理](#)」を参照してください。

例えば、アカウント 444455556666 にアカウント 111122223333 の対称暗号化 KMS キーの使用を許可するとします。これを行うには、次の例のようなポリシーステートメントを、アカウント 111122223333 の KMS キーのキーポリシーに追加します。このポリシーステートメントは、対称暗号化 KMS キーの暗号化オペレーションで KMS キーを使用する許可を外部アカウントである 444455556666 に付与します。

Note

次の例は、KMS キーを別のアカウントと共有するためのサンプルキーポリシーを示しています。例の Sid、Principal、Action の値を、KMS キーの使用目的に合った有効な値に置き換えます。

```
{
  "Sid": "Allow an external account to use this KMS key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::444455556666:root"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

外部アカウントにアクセス許可を付与する代わりに、キーポリシーで特定の外部ユーザーとロールを指定できます。ただし、外部アカウントの IAM 管理者が適切な IAM ポリシーを IAM 管理者のアイデンティティにアタッチするまで、これらのユーザーとロールは KMS キーを使用できません。IAM ポリシーは、キーポリシーで指定されている外部ユーザーとロールのすべてまたはサブセットにアクセス許可を与えることができます。また、キーポリシーで指定されたアクションのすべてまたはサブセットを許可できます。

キーポリシーで ID を指定すると、外部アカウントの IAM 管理者が提供できるアクセス権限が制限されます。ただし、2 つのアカウントを使用したポリシー管理はより複雑になります。例えば、ユーザーまたはロールを追加する必要があるとします。IAM 管理者のアイデンティティを、KMS キーを所有するアカウントのキーポリシーに追加し、アイデンティティのアカウントに IAM ポリシーを作成する必要があります。

キーポリシーで特定の外部ユーザーまたはロールを指定するには、Principal 要素に、外部アカウントのユーザーまたはロールの Amazon リソースネーム (ARN) を入力します。

例えば、次のキーポリシーステートメントの例では、アカウント 444455556666 の ExampleRole に、アカウント 111122223333 の KMS キーの使用を許可します。このキーポリシーステートメントは、対称暗号化 KMS キーの暗号化オペレーションで KMS キーを使用する許可を外部アカウントである 444455556666 に付与します。

Note

次の例は、KMS キーを別のアカウントと共有するためのサンプルキーポリシーを示しています。例の Sid、Principal、Action の値を、KMS キーの使用目的に合った有効な値に置き換えます。

```
{
  "Sid": "Allow an external account to use this KMS key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::444455556666:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

Note

[条件](#)を使用してキーポリシーを制限しない限り、アクセス許可を付与するキーポリシーステートメントで、プリンシパルをアスタリスク (*) に設定しないでください。アスタリスクは、別のポリシーステートメントが明示的に拒否しない限り、KMS キーを使用するすべての AWS アカウント アクセス許可をすべての のすべてのアイデンティティに付与します。他のユーザーは AWS アカウント、自分のアカウントで対応するアクセス許可を持っている場合はいつでも KMS キーを使用できます。

また、外部アカウントに付与するアクセス許可を決定する必要があります。例えば、復号するが暗号化しないアクセス許可をユーザーに付与したり、KMS キーが表示されても使用できないアクセス許可を付与したりできます。KMS キーに対するアクセス許可のリストについては、[AWS KMS アクセス許可](#) を参照してください。

KMS キーを作成する際のキーポリシーの設定

[CreateKey](#) オペレーションを使用して KMS キーを作成する場合、その Policy パラメータを使用して、KMS キー使用許可を外部アカウント、または外部ユーザーとロールに付与するキーポリシーを指定できます。

で KMS キーを作成するときは AWS Management Console、そのキーポリシーも作成します。キー管理者セクションとキーユーザーセクションでアイデンティティを選択すると、AWS KMS はこれらのアイデンティティのポリシーステートメントを KMS キーのキーポリシーに追加します。Key Users セクションでは、外部アカウントをキーユーザーとして追加することもできます。

外部アカウントのアカウント ID を入力すると、はキーポリシーに 2 つのステートメント AWS KMS を追加します。このアクションは、キーポリシーにのみ影響します。外部アカウントのユーザーとロールは、IAM ポリシーをアタッチしてこれらのアクセス許可の一部またはすべてを付与するまで、KMS キーを使用できません。

最初のポリシーステートメントでは、暗号化オペレーションで KMS キーを使用するアクセス許可を外部アカウントに付与します。2 番目のキーポリシーステートメントでは、外部アカウントが KMS キーに対する許可を作成、表示、および取り消すことを許可します。ただし、リクエスト [AWS が統合されている サービス AWS KMS](#) から送信された場合に限り。これらのアクセス許可により、ユーザーデータを暗号化する他の AWS のサービスが KMS キーを使用できるようになります。これらのアクセス許可は、AWS サービス内でユーザーデータを暗号化する KMS キー向けに設計されています。

ステップ 2: 外部アカウントに IAM ポリシーを追加する

KMS キーを所有するアカウントのキーポリシーは、アクセス許可の有効範囲を設定します。ただし、外部アカウントのユーザーとロールは、それらのアクセス許可を委任する IAM ポリシーをアタッチするか、権限を使用して KMS キーへのアクセスを管理するまで、KMS キーを使用できません。IAM ポリシーは外部アカウントで設定されます。

キーポリシーが外部アカウントにアクセス許可を与える場合は、アカウント内の任意のユーザーまたはロールに IAM ポリシーをアタッチできます。ただし、キーポリシーが指定したユーザーまたはロールにアクセス許可を付与する場合、IAM ポリシーでは、指定したユーザーとロールのすべてまたはサブセットにのみそれらのアクセス許可を付与できます。IAM ポリシーが他の外部ユーザーまたはロールに KMS キーへのアクセスを許可しても、影響はありません。

キーポリシーは、IAM ポリシー内のアクションも制限します。IAM ポリシーは、キーポリシーで指定されたアクションのすべてまたはサブセットを委任できます。IAM ポリシーに、キーポリシーで指定されていないアクションがリストされている場合、それらのアクセス権限は有効ではありません。

以下の IAM ポリシー例では、プリンシパルがアカウント 111122223333 の KMS キーを暗号化オペレーションに使用することを許可します。アカウント内のユーザーとロールにこの権限を付与するには 444455556666、アカウント内のユーザーまたはロールに[ポリシーをアタッチ](#) 444455556666します。

 Note

次の例は、KMS キーを別のアカウントと共有するための IAM ポリシーのサンプルを示しています。例の Sid、Resource、Action の値を、KMS キーの使用目的に合った有効な値に置き換えます。

```
{
  "Sid": "AllowUseOfKeyInAccount111122223333",
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

このポリシーに関する以下の詳細情報に注意してください。

- キーポリシーとは異なり、IAM ポリシーステートメントには Principal 要素が含まれていません。IAM ポリシーでは、プリンシパルはポリシーがアタッチされている ID です。
- IAM ポリシーの Resource エレメントは、プリンシパルが使用できる KMS キーを識別します。KMS キーを指定するには、その[キー ARN](#) を Resource 要素に追加します。

- Resource 要素には複数の KMS キーを指定できます。ただし、Resource 要素で特定の KMS キーを指定しないと、意図したよりも多くの KMS キーへのアクセス許可を、誤って付与する可能性があります。
- 外部ユーザーが [AWS KMSと統合されたAWS サービス](#) で KMS キーを使用できるようにするには、キーポリシーまたは IAM ポリシーにアクセス許可を追加する必要があります。詳細については、「[外部 KMS キーの使用を許可する AWS のサービス](#)」を参照してください。

IAM ポリシーのオペレーションの詳細については、「[IAM ポリシー](#)」を参照してください。

外部 KMS キーの使用を許可する AWS のサービス

別のアカウントのユーザーに、AWS KMSと統合されたサービスで KMS キーを使用するアクセス許可を付与できます。例えば、外部アカウントのユーザーが KMS キーを使用して Amazon S3 バケット内のオブジェクトを暗号化したり、AWS Secrets Managerに保存されたシークレットを暗号化することができます。

キーポリシーでは、KMS キーを使用するための許可を外部ユーザーまたは外部ユーザーのアカウントに付与する必要があります。さらに、AWS のサービスを使用する許可をユーザーに付与するアイデンティティに、IAM ポリシーをアタッチする必要があります。また、このサービスでは、ユーザーがキーポリシーまたは IAM ポリシーで、追加のアクセス許可を持っている必要があります。カスターマネージドキーで AWS のサービス 必要とするアクセス許可のリストについては、サービスのユーザーガイドまたはデベロッパーガイドの「セキュリティ」の章の「データ保護」トピックを参照してください。

他のアカウントで KMS キーを使用する

別の で KMS キーを使用するアクセス許可がある場合は AWS アカウント、AWS Management Console、AWS SDKs AWS CLI、および で KMS キーを使用できます AWS Tools for PowerShell。

シェルコマンドまたは API リクエストで別のアカウントの KMS キーを識別するには、次の[キー識別子](#)を使用します。

- [暗号化オペレーション](#)、[DescribeKey](#)、[GetPublicKey](#) の場合は、KMS キーの[キー ARN](#) または[エイリアス ARN](#) を使用します。
- [CreateGrant](#)、[GetKeyRotationStatus](#)、[ListGrants](#)、[RevokeGrant](#) の場合、KMS キーのキー ARN を使用します。

キー ID またはエイリアス名のみを入力すると、は KMS キーがアカウントにある AWS と仮定します。

AWS KMS コンソールには、KMS キーを使用するアクセス許可があっても、他のアカウントには KMS キーが表示されません。他の AWS サービスのコンソールに表示される KMS キーのリストには、他のアカウントの KMS キーは含まれません。

AWS サービスのコンソールで別のアカウントで KMS キーを指定するには、KMS キーのキー ARN またはエイリアス ARN を入力する必要があります。必要なキー識別子はサービスによって異なり、サービスコンソールとその API オペレーションとの間でも異なる場合があります。詳細については、サービスのドキュメントを参照してください。

マルチリージョンキーへのアクセスを制御する

マルチリージョンキーは、単一リージョンキーを使用するとより複雑な、コンプライアンス、災害対策、バックアップのシナリオで使用できます。ただし、マルチリージョンキーのセキュリティプロパティは単一リージョンキーのセキュリティプロパティとは大きく異なるため、マルチリージョンキーの作成、管理、使用の認可には注意が必要です。

Note

Resource フィールドのワイルドカード文字を含む既存の IAM ポリシーステートメントが、単一リージョンキーおよびマルチリージョンキーの両方に適用されるようになりました。ステートメントを単一リージョンの KMS キーまたはマルチリージョンキーに制限するには、[kms:MultiRegion](#) 条件キーを使用します。

認可ツールを使用して、単一リージョンで十分なシナリオでのマルチリージョンキーの作成および使用を阻止します。マルチリージョンキーを必要とする ののみレプリケート AWS リージョン することをプリンシパルに許可します。マルチリージョンキーのアクセス許可を、それらを必要とするプリンシパルおよびタスクに対してのみ付与します。

キーポリシー、IAM ポリシー、許可を使用して、IAM プリンシパルが でマルチリージョンキーを管理および使用できるようにします AWS アカウント。各マルチリージョンキーは、一意のキー ARN とキーポリシーを持つ独立したリソースです。各キーのキーポリシーを確立して維持し、新規および既存の IAM ポリシーが認可戦略を実装していることを確認する必要があります。

マルチリージョンキーをサポートするために、は IAM サービスにリンクされたロール AWS KMS を使用します。このロールは、AWS KMS に[共有プロパティ](#)の同期に必要なアクセス許可を付与し

ます。詳細については、「[マルチリージョンキー AWS KMS の同期を許可する](#)」を参照してください。

トピック

- [マルチリージョンキーの認可の原則](#)
- [マルチリージョンキー管理者およびユーザーを認可する](#)

マルチリージョンキーの認可の原則

マルチリージョンキーのキーポリシーと IAM ポリシーを設計するときは、次の原則を考慮します。

- キーポリシー — 各マルチリージョンキーは、固有の[キーポリシー](#)を持つ独立した KMS キーリソースです。関連するマルチリージョンキーセットの各キーに、同じ、または異なるキーポリシーを適用できます。キーポリシーは、マルチリージョンキーの[共有プロパティ](#)ではありません。AWS KMS は、関連するマルチリージョンキー間でキーポリシーをコピーまたは同期しません。

AWS KMS コンソールでレプリカキーを作成すると、コンソールにはプライマリキーの現在のキーポリシーが便利に表示されます。このキーポリシーの使用、編集、削除、置き換えを行うことができます。ただし、プライマリキーポリシーを変更せずに受け入れた場合でも、AWS KMS はポリシーを同期しません。例えば、プライマリキーのキーポリシーを変更しても、レプリカキーのキーポリシーは変わりません。

- デフォルトのキーポリシー — [CreateKey](#) および [ReplicateKey](#) オペレーションを使用してマルチリージョンキーを作成する際、リクエストでキーポリシーを指定しない限り、[デフォルトのキーポリシー](#)が適用されます。これは、単一リージョンキーに適用されるのと同じデフォルトのキーポリシーです。
- IAM ポリシー — すべての KMS キーと同様に、[キーポリシーによって許可される](#)場合にのみ、IAM ポリシーを使用してマルチリージョンキーへのアクセスを制御できます。[IAM ポリシー](#)は AWS リージョン、デフォルトですべてのリージョンに適用されます。ただし、条件キー ([aws:RequestedRegion](#) など) を使用して、特定のリージョンへのアクセス許可を制限できます。

プライマリキーおよびレプリカキーを作成するには、キーが作成されたリージョンに適用される、IAM ポリシーの `kms:CreateKey` アクセス許可をプリンシパルに付与する必要があります。

- グラント — AWS KMS [グラント](#)はリージョン別です。各権限は、1 つの KMS キーにアクセス許可を付与します。権限を使用して、マルチリージョンのプライマリキーまたはレプリカキーへのアクセス許可を付与できます。ただし、マルチリージョンキーが関連付けられている場合でも、単一の権限を使用して複数の KMS キーにアクセス許可を付与することはできません。

- キー ARN — 各マルチリージョンキーは [一意のキー ARN](#) を持ちます。関連するマルチリージョンキーのキー ARN は、同じパーティション、アカウント、キー ID を持ちますが、リージョンが異なります。

IAM ポリシーステートメントを特定のマルチリージョンキーに適用するには、そのキー ARN またはリージョンを含むキー ARN パターンを使用します。関連するすべてのマルチリージョンキーに IAM ポリシーステートメントを適用するには、次の例に示すように、ARN のリージョン要素でワイルドカード文字 (*) を使用します。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Describe*",
    "kms:List*"
  ],
  "Resource": {
    "arn:aws:kms:*::111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab"
  }
}
```

ポリシーステートメントを 内のすべてのマルチリージョンキーに適用するには AWS アカウント、[kms:MultiRegion](#) ポリシー条件または固有のmrk-プレフィックスを含むキー ID パターンを使用できます。

- サービスリンクロール — マルチリージョンのプライマリキーを作成するプリンシパルには、[iam:CreateServiceLinkedRole](#) アクセス許可が必要です。

関連するマルチリージョンキーの共有プロパティを同期するために、AWS KMS は IAM [サービスにリンクされたロール](#) を引き受けます。マルチリージョンのプライマリキー AWS KMS を作成するたびに、AWS アカウント でサービスにリンクされたロールが作成されます。(ロールが存在する場合は、AWS KMS は悪影響のないロールを再作成します)。ロールはすべてのリージョンで有効です。AWS KMS がサービスにリンクされたロールを作成 (または再作成) できるようにするには、マルチリージョンのプライマリキーを作成するプリンシパルに [iam:CreateServiceLinkedRole](#) アクセス許可が必要です。

マルチリージョンキー管理者およびユーザーを認可する

マルチリージョンキーを作成および管理するプリンシパルには、プライマリリージョンとレプリカリージョンで次のアクセス許可が必要です。

- kms:CreateKey
- kms:ReplicateKey
- kms:UpdatePrimaryRegion
- iam:CreateServiceLinkedRole

プライマリキーを作成する

[マルチリージョンのプライマリキーを作成](#)するには、プリンシパルに、プライマリキーのリージョンで有効な IAM ポリシーの [kms:CreateKey](#) および [iam:CreateServiceLinkedRole](#) アクセス許可が必要です。これらのアクセス許可を持つプリンシパルは、アクセス許可が制限されない限り 単一リージョンキーおよびマルチリージョンキーを作成できます。

アクセスiam:CreateServiceLinkedRole許可により、AWS KMS は [AWSServiceRoleForKeyManagementServiceMultiRegionKeys](#) [ロール](#)を作成して、関連するマルチリージョンキーの[共有プロパティ](#)を同期できます。

例えば、この IAM ポリシーはプリンシパルに、任意のタイプの KMS キーの作成を許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Action": [
      "kms:CreateKey",
      "iam:CreateServiceLinkedRole"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
}
```

マルチリージョンのプライマリキーを作成するアクセス許可を許可または拒否するには、[kms:MultiRegion](#) 条件キーを使用します。有効な値は、true (マルチリージョンキー) または false (単一リージョンキー) です。例えば、次の IAM ポリシーステートメントでは、kms:MultiRegion 条件キーを持つ Deny アクションを使用して、プリンシパルがマルチリージョンキーを作成しないようにします。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Action": "kms:CreateKey",
    "Effect": "Deny",
    "Resource": "*",
    "Condition": {
      "Bool": {
        "kms:MultiRegion": true
      }
    }
  }
}
```

キーをレプリケートする

[マルチリージョンのレプリカキーを作成する](#)には、プリンシパルに次のアクセス許可が必要です。

- プライマリキーのキーポリシーの [kms:ReplicateKey](#) アクセス許可。
- レプリカキーリージョンで有効な IAM ポリシーの [kms:CreateKey](#) アクセス許可。

これらのアクセス許可を許可する場合は注意が必要です。これにより、プリンシパルは KMS キーと、その使用を認可するキーポリシーを作成できます。kms:ReplicateKey アクセス許可は AWS KMS内のリージョンの境界を越えるキーマテリアルの転送も認可します。

マルチリージョンキー AWS リージョン をレプリケートできる を制限するには、[kms:ReplicaRegion](#) 条件キーを使用します。これは、kms:ReplicateKey アクセス許可のみを制限します。それ以外には影響を与えません。例えば、次のキーポリシーは、指定されたリージョンでのみプリンシパルにこのプライマリキーのレプリケーションを許可します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
  "Action": "kms:ReplicateKey",
  "Resource": "*",
}
```

```
"Condition": {
  "StringEquals": {
    "kms:ReplicaRegion": [
      "us-east-1",
      "eu-west-3",
      "ap-southeast-2"
    ]
  }
}
```

プライマリリージョンを更新する

認可されたプリンシパルは、レプリカキーをプライマリキーに変換し、プライマリキーは以前のプライマリキーをレプリカキーに変更します。このアクションは[プライマリリージョンの更新](#)として知られています。。プライマリリージョンを更新するには、プリンシパルに両方のリージョンで[kms:UpdatePrimaryRegion](#) 許可が必要です。キーポリシーまたは IAM ポリシーでこれらのアクセス許可を付与できます。

- プライマリキーの kms:UpdatePrimaryRegion。このアクセス許可は、プライマリキーリージョンで有効である必要があります。
- レプリカキーの kms:UpdatePrimaryRegion。このアクセス許可は、レプリカキーリージョンで有効である必要があります。

例えば、次のキーポリシーは、KMS キーのプライマリリージョンを更新する管理者のロールを引き受けることができるユーザーに付与します。この KMS キーを、このオペレーションでプライマリキーまたはレプリカキーにすることができます。

```
{
  "Effect": "Allow",
  "Resource": "*",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Administrator"
  },
  "Action": "kms:UpdatePrimaryRegion"
}
```

プライマリキーをホスト AWS リージョン できる を制限するには、[kms:PrimaryRegion](#) 条件キーを使用します。たとえば、次の IAM ポリシーステートメントでは、新しいプライマリリージョンが指

定されたリージョンの 1 つである場合にのみ AWS アカウント、プリンシパルが のマルチリージョンキーのプライマリリージョンを更新することを許可します。

```
{
  "Effect": "Allow",
  "Action": "kms:UpdatePrimaryRegion",
  "Resource": {
    "arn:aws:kms:*:111122223333:key/*"
  },
  "Condition": {
    "StringEquals": {
      "kms:PrimaryRegion": [
        "us-west-2",
        "sa-east-1",
        "ap-southeast-1"
      ]
    }
  }
}
```

マルチリージョンキーを使用および管理する

デフォルトでは、AWS アカウント で KMS キーを使用および管理するためのアクセス許可を持つプリンシパルは、マルチリージョンキーを使用および管理するためのアクセス許可も持っています。ただし、[kms:MultiRegion](#) 条件キーを使用して、単一リージョンキーのみまたはマルチリージョンキーのみを許可できます。または、[kms:MultiRegionKeyType](#) 条件キーを使用して、マルチリージョンのプライマリキーのみまたはレプリカキーのみを許可します。どちらの条件キーも、[CreateKey](#) オペレーションと、既存の KMS キーを使用するオペレーション ([Encrypt](#) または [EnableKey](#) など) へのアクセスを制御します。

以下の IAM ポリシーステートメントの例では、[kms:MultiRegion](#) 条件キーを使用して、プリンシパルがマルチリージョンキーを使用または管理できないようにします。

```
{
  "Effect": "Deny",
  "Action": "kms:*",
  "Resource": "*",
  "Condition": {
    "Bool": "kms:MultiRegion": true
  }
}
```

この IAM ポリシーステートメントの例では、`kms:MultiRegionKeyType` 条件を使用して、マルチリージョンレプリカキーのみでキー削除のスケジュールおよびキャンセルを実行することをプリンシパルに許可します。

```
{
  "Effect": "Allow",
  "Action": [
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": {
    "arn:aws:kms:us-west-2:111122223333:key/*"
  },
  "Condition": {
    "StringEquals": "kms:MultiRegionKeyType": "REPLICA"
  }
}
```

へのアクセスの確認 AWS KMS keys

に現在アクセスできるユーザーまたは対象を完全に特定するには AWS KMS key、KMS キーのキーポリシー、KMS キーに適用されるすべての[許可](#)、および潜在的にすべての AWS Identity and Access Management (IAM) ポリシーを調べる必要があります。これらを確認することで、潜在的な KMS キーの使用範囲を明らかにしたり、コンプライアンスや監査の要件を満たしたりできます。次のトピックは、現在 KMS キーにアクセスできる AWS プリンシパル (アイデンティティ) の完全なリストを作成するのに役立ちます。

トピック

- [キーポリシーを確認する](#)
- [IAM ポリシーの確認](#)
- [許可の確認](#)

キーポリシーを確認する

[キーポリシー](#)は、KMS キーへのアクセスを制御するための主要な方法です。すべての KMS キーには、厳密に 1 つのキーポリシーが必要です。

キーポリシーが[デフォルトのキーポリシー](#)で設定されるか、含まれている場合、キーポリシーによって、アカウントの IAM 管理者が IAM ポリシーを使用し、KMS キーへのアクセスを制御できるようになります。また、キーポリシーによって KMS キーを使用するアクセス許可が[別の AWS アカウント](#)に付与されている場合、外部アカウントの IAM 管理者は、IAM ポリシーを使用してこれらのアクセス許可を委任できます。KMS キーにアクセスできるプリンシパルの完全なリストを確認するには、[IAM ポリシーを調べます](#)。

アカウント[AWS マネージドキー](#)で AWS KMS [カスターマネージドキー](#)または のキーポリシーを表示するには、AWS KMS API で AWS Management Console または [GetKeyPolicy](#) オペレーションを使用します。キーポリシーを表示するには、KMS キーの `kms:GetKeyPolicy` アクセス許可が必要です。KMS キーのキーポリシーを表示する手順については、[the section called “キーポリシーを表示する”](#) を参照してください。

キーポリシードキュメントを確認し、各ポリシーステートメントの Principal 要素で指定されているすべてのプリンシパルを書き留めます。Allow 効果のあるポリシーステートメントでは、Principal 要素 AWS アカウント の IAM ユーザー、IAM ロール、および がこの KMS キーにアクセスできます。

Note

[条件](#)を使用してキーポリシーを制限しない限り、アクセス許可を付与するキーポリシーステートメントで、プリンシパルをアスタリスク (*) に設定しないでください。アスタリスクは、別のポリシーステートメントが明示的に拒否しない限り、KMS キーを使用するすべての AWS アカウント アクセス許可をすべての ID に付与します。他の のユーザーは AWS アカウント、自分のアカウントで対応するアクセス許可があるたびに KMS キーを使用できます。

次の例では、[デフォルトのキーポリシー](#)で見つかったポリシーステートメントを使用してこれを行う方法を示します。

Example ポリシーステートメント 1

```
{
  "Sid": "Enable IAM User Permissions",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
  "Action": "kms:*",
  "Resource": "*"
}
```

ポリシーステートメント 1 では、`arn:aws:iam::111122223333:root`は AWS アカウント 111122223333 を参照する[AWS アカウントプリンシパル](#)です。(アカウントのルートユーザーではありません)。デフォルトでは、新しい KMS キーを作成するとき AWS Management Console、またはプログラムで新しい KMS キーを作成するときに、このようなポリシーステートメントがキーポリシードキュメントに含まれますが、キーポリシーは提供されません。

へのアクセスを許可するステートメントを含むキーポリシードキュメント AWS アカウント は、[アカウントの IAM ポリシーが KMS キーへのアクセスを許可できるようにします](#)。つまり、アカウントのユーザーとロールは、キーポリシードキュメントにプリンシパルとして明示的にリストされていない場合でも、KMS キーにアクセスできる可能性があります。プリンシパルとして AWS アカウント リストされているすべての のすべての [IAM ポリシーを調べて](#)、この KMS キーへのアクセスを許可するかどうかを確認してください。

Example ポリシーステートメント 2

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/KMSKeyAdmins"},
  "Action": [
    "kms:Describe*",
    "kms:Put*",
    "kms:Create*",
    "kms:Update*",
    "kms:Enable*",
    "kms:Revoke*",
    "kms:List*",
    "kms:Disable*",
    "kms:Get*",
    "kms>Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

ポリシーステートメント 2 では、`KMSKeyAdmins` in AWS アカウント 111122223333 という名前の IAM ロール `arn:aws:iam::111122223333:role/KMSKeyAdmins`を参照します。このロールを引き受ける権限を持つユーザーは、KMS キーを管理するための管理アクションである、ポリシーステートメントにリストされたアクションを実行できます。

Example ポリシーステートメント 3

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/EncryptionApp"},
  "Action": [
    "kms:DescribeKey",
    "kms:GenerateDataKey*",
    "kms:Encrypt",
    "kms:ReEncrypt*",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

ポリシーステートメント 3 では、は EncryptionApp in AWS アカウント 111122223333 という名前の IAM ロールarn:aws:iam::111122223333:role/EncryptionAppを参照します。このロールを引き受ける権限を持つプリンシパルは、対称暗号化 KMS キーの[暗号化オペレーション](#)を含む、ポリシーステートメントにリストされたアクションを実行できます。

Example ポリシーステートメント 4

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:role/EncryptionApp"},
  "Action": [
    "kms:ListGrants",
    "kms:CreateGrant",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

ポリシーステートメント 4 では、は EncryptionApp in AWS アカウント 111122223333 という名前の IAM ロールarn:aws:iam::111122223333:role/EncryptionAppを参照します。このロールを引き受ける権限を持つプリンシパルは、ポリシーステートメントにリストされたアクションを実行できます。これらのアクションは、ポリシーステートメント 3 の例で許可されたアクションと連動する際に、KMS キーの使用を、[AWS KMSと統合されたほとんどのAWS のサービス](#) (特

に、[権限](#)を使用するサービス) に委任するために必要なアクションになります。Condition 要素の [kms:GrantIsForAWSResource](#) 値により、委任がと統合 AWS KMS され、認可に許可を使用する AWS サービスである場合にのみ、委任が許可されます。

キーポリシードキュメントでプリンシパルを指定する各種方法をすべて確認するには、IAM ユーザーガイドの[プリンシパルの指定](#)を参照してください。

AWS KMS キーポリシーの詳細については、「」を参照してください[のキーポリシー AWS KMS](#)。

IAM ポリシーの確認

キーポリシーと許可に加え、[IAM ポリシー](#)を使用して KMS キーへのアクセスを許可することもできます。IAM ポリシーとキーポリシーがどのように連携するかについては、「[アクセス AWS KMS 許可のトラブルシューティング](#)」を参照してください。

IAM ポリシーを使用して KMS キーに現在アクセスできるプリンシパルを特定するには、ブラウザベースの [IAM Policy Simulator](#) ツールを使用するか、IAM API にリクエストします。

IAM ポリシーを調べる方法

- [IAM ポリシーシミュレーターを使用した IAM ポリシーの確認](#)
- [IAM API を使用した IAM ポリシーの確認](#)

IAM ポリシーシミュレーターを使用した IAM ポリシーの確認

IAM Policy Simulator は、IAM ポリシーを介して KMS キーにアクセスできるプリンシパルを学習するのに役立ちます。

IAM Policy Simulator を使用して KMS キーへのアクセスを特定するには

1. にサインインし AWS Management Console 、 で IAM Policy Simulator を開きます<https://policysim.aws.amazon.com/>。
2. [Users, Groups, and Roles] ペインで、ポリシーをシミュレートするユーザー、グループ、またはロールを選択します。
3. (オプション) シミュレーションから除外するポリシーの横のチェックボックスをオフにします。すべてのポリシーをシミュレートする場合は、すべてのポリシーが選択された状態にします。
4. [Policy Simulator] ペインで、以下のオペレーションを行います。
 - a. [Select service] で、[Key Management Service] を選択します。

- b. 特定の AWS KMS アクションをシミュレートするには、アクションの選択で、シミュレートするアクションを選択します。すべての AWS KMS アクションをシミュレートするには、すべて選択を選択します。
5. (オプション) Policy Simulator が、デフォルトですべての KMS キーへのアクセスをシミュレートします。特定の KMS キーへのアクセスをシミュレートするには、[Simulation Settings] (シミュレーション設定) を選択し、シミュレートする KMS キーの Amazon リソースネーム (ARN) を入力します。
6. [Run Simulation (シミュレーションの実行)] を選択します。

シミュレーションの結果は、[Results] セクションに表示されます。AWS アカウントのすべてのユーザー、グループ、ロールについて、ステップ 2~6 を繰り返します。

IAM API を使用した IAM ポリシーの確認

IAM API を使用して、IAM ポリシーをプログラムで調べることができます。次のステップは、API でユーザーベースのポリシーを確認する方法の概要を示します。

1. キーポリシーにプリンシパルとして AWS アカウント リストされている各 (つまり、この形式で指定された各 [AWS アカウントプリンシパル](#): "Principal": {"AWS": "arn:aws:iam::111122223333:root"}) について、IAM API の [ListUsers](#) および [ListRoles](#) オペレーションを使用して、アカウントのすべてのユーザーとロールを取得します。
2. リスト内の各ユーザーとロールについて、IAM API の [SimulatePrincipalPolicy](#) オペレーションを使用し、以下のパラメータを渡します。
 - PolicySourceArn について、リストのユーザーやロールから Amazon リソースネーム (ARN) を指定します。PolicySourceArn は、各 SimulatePrincipalPolicy リクエストで 1 つしか指定できないため、このオペレーションは複数回 (リスト内のユーザーおよびロールごとに 1 回) 呼び出す必要があります。
 - ActionNames リストには、シミュレートするすべての AWS KMS API アクションを指定します。すべての AWS KMS API アクションをシミュレートするには、`kms:*` を使用します。`kms:*`。個々の AWS KMS API アクションをテストするには、各 API アクションの前に `kms:`「」のように「」を付け `kms:ListKeys` ます。AWS KMS API アクションの完全なリストについては、「AWS Key Management Service API リファレンス」の「[Actions](#)」(アクション) を参照してください。
 - (オプション) ユーザーやロールが特定の KMS キーにアクセスできるかどうかを特定するために、ResourceArns パラメータを使用して KMS キーの Amazon リソースネーム (ARN) のリス

トを指定します。ユーザーまたはロールが KMS キーにアクセスできるかどうかを特定するために、ResourceArns パラメータを省略してください。

IAM は各 SimulatePrincipalPolicy リクエストに対して、allowed, explicitDeny,、またはの評価決定で応答 implicitDenyします。の評価決定を含むレスポンスごとにallowed、レスポンスには許可される特定の AWS KMS API オペレーションの名前が含まれます。評価で使った KMS キーの ARN がある場合は、これも含まれます。

許可の確認

許可は、ユーザーまたはと統合された AWS のサービスが AWS KMS KMS キーを使用する方法とタイミングを指定するために使用できるアクセス許可を指定する高度なメカニズムです。権限は KMS キーにアタッチされ、各権限には KMS キーを使用するためのアクセス許可を受け取るプリンシパルと、許可されるオペレーションのリストが含まれます。許可は、キーポリシーに代わる手段であり、特定のユースケースで役立ちます。詳細については、「[の許可 AWS KMS](#)」を参照してください。

KMS キーの許可のリストを取得するには、AWS KMS [ListGrants](#) オペレーションを使用します。KMS キーの権限を確認することで、これらの権限で KMS キーを使用するアクセス許可を現在持っているユーザーやアプリケーションを特定できます。例えば、AWS CLIの [list-grants](#) コマンドから取得した権限の JSON 表現を次に示します。

```
{"Grants": [{
  "Operations": ["Decrypt"],
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Name": "0d8aa621-43ef-4657-b29c-3752c41dc132",
  "RetiringPrincipal": "arn:aws:iam::123456789012:root",
  "GranteePrincipal": "arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/i-5d476fab",
  "GrantId": "dc716f53c93acacf291b1540de3e5a232b76256c83b2ecb22cdefa26576a2d3e",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "CreationDate": 1.444151834E9,
  "Constraints": {"EncryptionContextSubset": {"aws:ebs:id": "vol-5cccfb4e"}}
}]}
```

KMS キーにアクセスできるユーザーやアプリケーションを見つけるには、"GranteePrincipal" 要素を確認します。前述の例では、被付与者のプリンシパルは、EC2 インスタンス i-5d476fab に関連して割り当てられたロールユーザーです。EC2 インフラストラクチャはこのロールを使用して、暗号化された EBS ボリューム vol-5cccfb4e をインスタンスにアタッチします。この場合、EC2 イン

フラストラクチャロールは、KMS キーを使用するアクセス許可を持っています。これは、この KMS キーで保護された、暗号化された EBS ボリュームを以前に作成したためです。その後、ボリュームを EC2 インスタンスにアタッチしました。

以下は、AWS CLIの [list-grants](#) コマンドから取得した権限の JSON 表現の別の例です。次の例では、被付与者プリンシパルは別のプリンシパルです AWS アカウント。

```
{"Grants": [{
  "Operations": ["Encrypt"],
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Name": "",
  "GranteePrincipal": "arn:aws:iam::444455556666:root",
  "GrantId": "f271e8328717f8bde5d03f4981f06a6b3fc18bcae2da12ac38bd9186e7925d11",
  "IssuingAccount": "arn:aws:iam::111122223333:root",
  "CreationDate": 1.444151269E9
}]}
```

暗号化コンテキスト

Note

[非対称 KMS キー](#)または [HMAC KMS キー](#)を使用する暗号化オペレーションで暗号化コンテキストを指定することはできません。非対称アルゴリズムと MAC アルゴリズムは、暗号化コンテキストをサポートしません。

[対称暗号化 KMS キー](#)を使用したすべての AWS KMS [暗号化オペレーション](#)は、暗号化コンテキストを受け入れます。これは、データに関する追加のコンテキスト情報を含むことができるシークレット以外のキーと値のペアのオプションセットです。の Encrypt オペレーションに暗号化コンテキストを挿入 AWS KMS して、AWS KMS API 復号呼び出しの認可と監査可能性を高めることができます。AWS KMS は、暗号化コンテキストを追加の認証データ (AAD) として使用して、認証された暗号化をサポートします。暗号化コンテキストは、暗号化されて暗号文にバインドされます。これにより、データの復号には同じ暗号化コンテキストが必要になります。

この暗号化コンテキストは秘密ではなく、暗号化されていません。これは [AWS CloudTrail ログ](#)にプレーンテキストで表示されるため、それを使用して暗号化オペレーションを識別して分類できます。暗号化コンテキストには機密情報を含めないようにしてください。暗号化または復号されるデータの

内容を説明した暗号化テキストの使用をお勧めします。例えば、ファイルを暗号化するときは、ファイルパスの一部を暗号化コンテキストとして使用することもあります。

```
"encryptionContext": {
  "department": "10103.0"
}
```

例えば、[Amazon Elastic Block Store](#) (Amazon EBS) [CreateSnapshot](#) オペレーションで作成されたボリュームとスナップショットを暗号化する際、Amazon EBS はボリューム ID を暗号化コンテキスト値として使用します。

```
"encryptionContext": {
  "aws:ebs:id": "vol-abcde12345abc1234"
}
```

暗号化コンテキストを使用して、アカウント AWS KMS keys 内の へのアクセスを絞り込んだり制限したりすることもできます。暗号化コンテキストは[グラントの制約として](#)、および[ポリシーステートメントの条件](#)として使用できます。

暗号化コンテキストを使用して暗号化されたデータの整合性を保護する方法については、AWS セキュリティブログの「[AWS Key Management Service および EncryptionContext を使用して暗号化されたデータの整合性を保護する方法](#)」を参照してください。

暗号化コンテキストのルール

AWS KMS は、暗号化コンテキストのキーと値に次のルールを適用します。

- 暗号化コンテキストペアのキーと値はシンプルなりテラル文字列であることが必要です。整数や浮動小数点数など別の型を使用する場合、AWS KMS では文字列として解釈されます。
- 暗号化コンテキストのキーと値には、Unicode 文字を含めることができます。暗号化コンテキストにキーポリシーまたは IAM ポリシーで許可されていない文字が含まれている場合は、[kms:EncryptionContext:context-key](#) および [kms:EncryptionContextKeys](#) などのポリシー条件キーで暗号化コンテキストを指定できなくなります。キーポリシードキュメントのルールに関する詳細については、「[キーポリシー形式](#)」を参照してください。IAM ポリシードキュメントのルールに関する詳細については、「IAM ユーザーガイド」の「[IAM 名前の要件](#)」を参照してください。

ポリシーでの暗号化コンテキスト

暗号化コンテキストは主に整合性と信頼性を検証するために使用されます。ただし、キーポリシーおよび IAM ポリシーで、対称暗号化 AWS KMS keys へのアクセスを制御するための暗号化コンテキストを使用することも可能です。

[KMS: EncryptionContext](#) および [KMS: EncryptionContextKeys](#) 条件キーでは、要求に特定の暗号化コンテキストキーまたはキーと値のペアが含まれている場合にのみ、アクセス許可を許可 (または拒否) します。

例えば、以下のキーポリシーステートメントでは、RoleForExampleApp ロールに Decrypt オペレーションでの KMS キーの使用を許可します。また、`kms:EncryptionContext:context-key` 条件キーを使用して、リクエストの暗号化コンテキストに `AppName:ExampleApp` 暗号化コンテキストペアが含まれる場合にのみ、このアクセス許可を付与します。

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/RoleForExampleApp"
  },
  "Action": "kms:Decrypt",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "kms:EncryptionContext:AppName": "ExampleApp"
    }
  }
}
```

これらの暗号化コンテキスト条件キーの詳細については、「[の条件キー AWS KMS](#)」を参照してください。

グラントでの暗号化コンテキスト

[グラントを作成する](#)ときに、[グラント](#)のアクセス許可の条件を確立するグラント制約を含めることができます。は 2 つのグラント制約 `EncryptionContextSubset` `EncryptionContextEquals` と AWS KMS をサポートします。どちらも暗号化オペレーションのリクエストに[暗号化コンテキスト](#)が含まれます。これらのグラント制約を使用するとき、グラントの許可は、暗号化オペレーションのリクエストで暗号化コンテキストがグラント制約の要件を満たしている場合にのみ有効になります。

例えば、`EncryptionContextEquals` のグラント制約を、[GenerateDataKey](#) オペレーションを許可するグラントに追加できます。この制約により、このグラントは、リクエストの暗号化コンテキストがグラント制約の暗号化コンテキストで大文字と小文字を区別して一致する場合にのみ、この操作を許可します。

```
$ aws kms create-grant \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --grantee-principal arn:aws:iam::111122223333:user/exampleUser \  
  --retiring-principal arn:aws:iam::111122223333:role/adminRole \  
  --operations GenerateDataKey \  
  --constraints EncryptionContextEquals={Purpose=Test}
```

被付与者プリンシパルからの次のようなリクエストは、`EncryptionContextEquals` の制約を満たします。

```
$ aws kms generate-data-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --key-spec AES_256 \  
  --encryption-context Purpose=Test
```

グラント制約の詳細については、「[グラントの制約の使用](#)」を参照してください。グラントの詳細については、「[the section called “権限”](#)」を参照してください。

暗号化コンテキストのログ記録

AWS KMS は AWS CloudTrail を使用して暗号化コンテキストをログに記録し、アクセスされた KMS キーとデータを判断できるようにします。ログエントリには、ログエントリ内の暗号化コンテキストによって参照された特定のデータの暗号化または復号に、どの KMS キーが使用されたかが正確に示されます。

Important

暗号化コンテキストは記録されるため、機密情報が含まれていてはなりません。

暗号化コンテキストの保存

[Decrypt](#) または [ReEncrypt](#) オペレーションを呼び出すときに暗号化コンテキストを簡単に使用できるように、暗号化コンテキストを暗号化されたデータとともに格納できます。暗号化や復号で必要

になったときに完全な暗号化コンテキストを作成できる十分な暗号化コンテキストのみを保存することをお勧めします。

例えば、暗号化コンテキストがファイルへの完全修飾パスである場合、そのパスの一部のみを、暗号化されたファイルの内容とともに保存します。次に、完全な暗号化コンテキストが必要になったら、保存されたフラグメントからコンテキストを再構築します。誰かがファイルを改ざんした場合 (名前の変更、別の場所への移動など)、暗号化コンテキスト値が変更され、復号リクエストは失敗します。

アクセス許可をテストする

を使用するには AWS KMS、AWS が API リクエストの認証に使用できる認証情報が必要です。認証情報には、KMS キーとエイリアスにアクセスするためのアクセス許可を含める必要があります。アクセス許可は、キーポリシー、IAM ポリシー、グラント、およびクロスアカウントアクセス制御によって決定されます。KMS キーへのアクセス制御に加えて、CloudHSM やカスタムキーストアへのアクセスを制御できます。

DryRun API パラメータを指定して、AWS KMS キーを使用するために必要なアクセス許可があることを確認できます。DryRun を使用して、AWS KMS API コールのリクエストパラメータが正しく指定されていることを確認することもできます。

トピック

- [DryRun パラメータとは](#)
- [API で DryRun を指定する](#)

DryRun パラメータとは

DryRun は、AWS KMS API 呼び出しが成功することを確認するために指定する API パラメータ (オプション) です。実際に AWS KMS を呼び出す前に、DryRun を使用して API 呼び出しをテストします。次のことを確認できます。

- AWS KMS キーを使用するために必要なアクセス許可があること。
- 呼び出しのパラメータが正しく指定されていること。

AWS KMS は、特定の API アクションで DryRun パラメータの使用をサポートします。

- [CreateGrant](#)

- [Decrypt](#)
- [DeriveSharedSecret](#)
- [暗号化](#)
- [GenerateDataKey](#)
- [GenerateDataKeyPair](#)
- [GenerateDataKeyPairWithoutPlaintext](#)
- [GenerateDataKeyWithoutPlaintext](#)
- [GenerateMac](#)
- [ReEncrypt](#)
- [RetireGrant](#)
- [RevokeGrant](#)
- [Sign](#)
- [検証](#)
- [VerifyMac](#)

DryRun パラメータを使用すると料金が発生し、標準の API リクエストとして課金されます。AWS KMS 料金の詳細については、[AWS Key Management Service 「料金表」](#)を参照してください。

DryRun パラメータを使用するすべての API リクエストは API のリクエストクォータに適用され、API リクエストクォータを超えた場合にスロットリング例外が発生する可能性があります。例えば、[Decrypt](#) を呼び出す際に DryRun を使用する場合でも DryRun を使用しない場合でも、同じ暗号化オペレーションクォータに対してカウントされます。詳細については、「[AWS KMS リクエストのスロットリング](#)」を参照してください。

AWS KMS API オペレーションへのすべての呼び出しはイベントとしてキャプチャされ、AWS CloudTrail ログに記録されます。DryRun パラメータを指定するすべてのオペレーションの出力が CloudTrail ログに表示されます。詳細については、「[を使用した AWS KMS API コールのログ記録 AWS CloudTrail](#)」を参照してください。

API で DryRun を指定する

を使用するにはDryRun、`-dry-run`パラメータをサポートするコマンドと AWS KMS API コールで AWS CLI パラメータを指定します。これを行うと、AWS KMS は呼び出しが成功するかどうかを検

証します。を使用する AWS KMS 呼び出し DryRun は常に失敗し、呼び出しが失敗した理由に関する情報を含むメッセージを返します。メッセージには次の例外が含まれます。

- `DryRunOperationException` - DryRun が指定されていなければリクエストは成功します。
- `ValidationException` - 間違った API パラメータが指定されたためリクエストが失敗しました。
- `AccessDeniedException` - KMS リソースで指定された API アクションを実行するアクセス許可がありません。

例えば、次のコマンドは [CreateGrant](#) オペレーションを使用し、keyUserRole ロールの継承が承認されたユーザーが指定の [対称 KMS キー](#) 上の [Decrypt](#) オペレーションを呼び出すことを許可するグラントを作成します。DryRun パラメータが指定されます。

```
$ aws kms create-grant \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --grantee-principal arn:aws:iam::111122223333:role/keyUserRole \  
  --operations Decrypt \  
  --dry-run
```

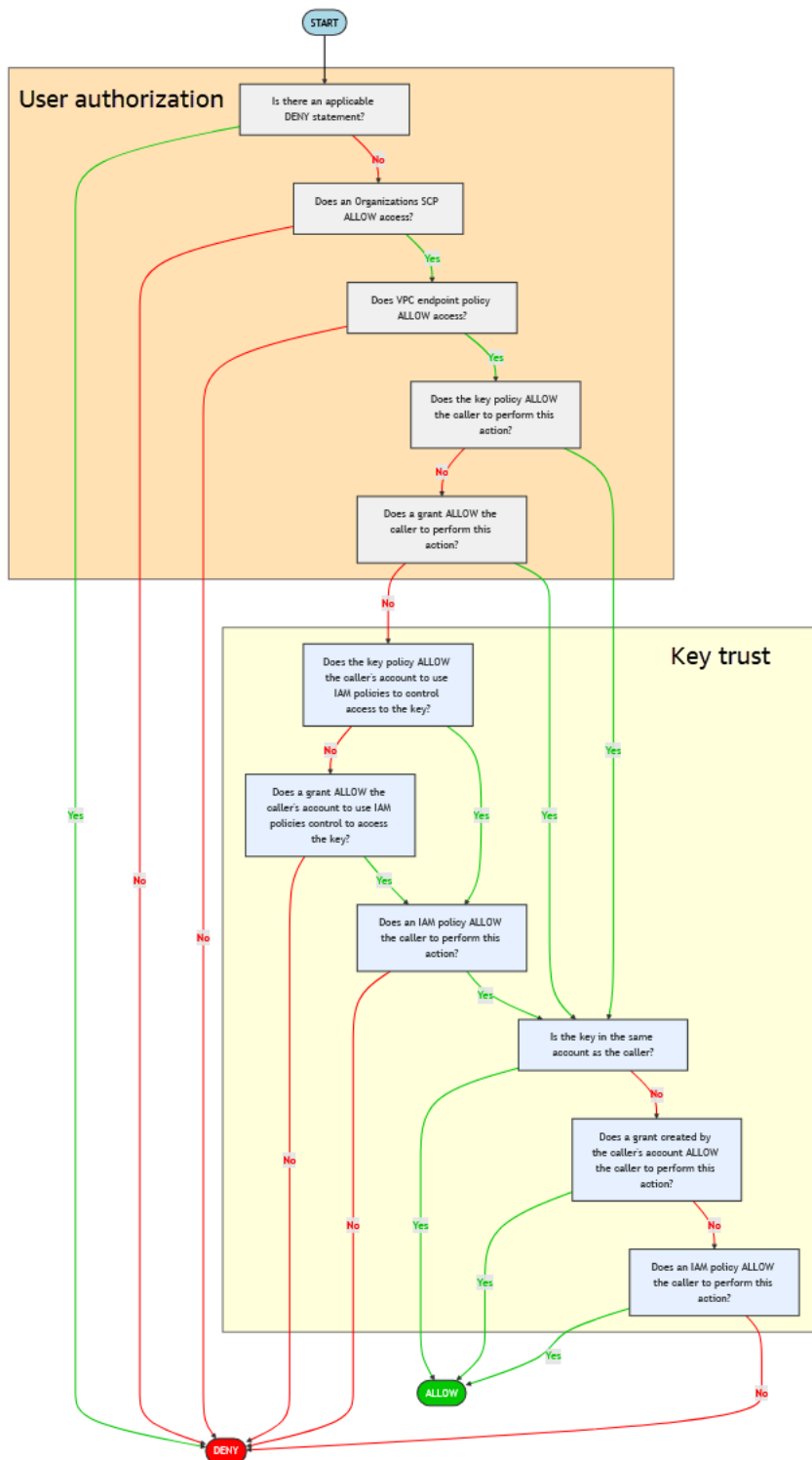
アクセス AWS KMS 許可のトラブルシューティング

KMS キーへのアクセスを許可すると、は以下 AWS KMS を評価します。

- KMS キーにアタッチされている [キーポリシー](#)。キーポリシーは常に、KMS キーを所有する AWS アカウント およびリージョンで定義されます。
- リクエストを行うユーザーまたはロールにアタッチされているすべての [IAM ポリシー](#)。プリンシパルの KMS キーの使用を管理する IAM ポリシーは、常にプリンシパルの AWS アカウントで定義されます。
- KMS キーに適用されるすべての [権限](#)。
- [AWS Organizations のサービスコントロールポリシー](#) や [VPC エンドポイントポリシー](#) など、KMS キーを使用するリクエストに適用できるその他の種類のポリシー。これらのポリシーはオプションであり、デフォルトですべてのアクションを許可しますが、プリンシパルに付与される権限を制限するために使用できます。

AWS KMS は、これらのポリシーメカニズムと一緒に評価して、KMS キーへのアクセスが許可されるか拒否されるかを判断します。これを行うには、次のフローチャートに示すようなプロセス AWS

KMS を使用します。以下のフローチャートは、ポリシーの評価プロセスを視覚的に表したものです。



このフローチャートは 2 つの部分に分かれています。これらの部分には順序があるように見えますが、一般的に同時に評価されます。

- 認可の使用は、キーポリシー、IAM ポリシー、権限、およびその他の適用可能なポリシーに基づいて、KMS キーの使用を許可するかどうかを判断します。
- キーの信頼は、使用が許可されている KMS キーを信頼すべきかどうかを判断します。一般に、のリソースを信頼します AWS アカウント。ただし、アカウントの権限または IAM ポリシーで KMS キーの使用が許可され AWS アカウント ている場合は、別の で KMS キーを使用することに自信を持つこともできます。

このフローチャートを使用すると、発信者が KMS キーの使用許可を許可または拒否された理由がわかります。また、ポリシーと許可を評価することもできます。例えば、フローチャートは、キーポリシー、IAM ポリシー、または許可で、明示的な DENY ステートメントまたは明示的な ALLOW ステートメントがないことによって、呼び出し元がアクセスを拒否できることを示しています。

フローチャートでは、いくつかの一般的なアクセス許可のシナリオについて説明しています。

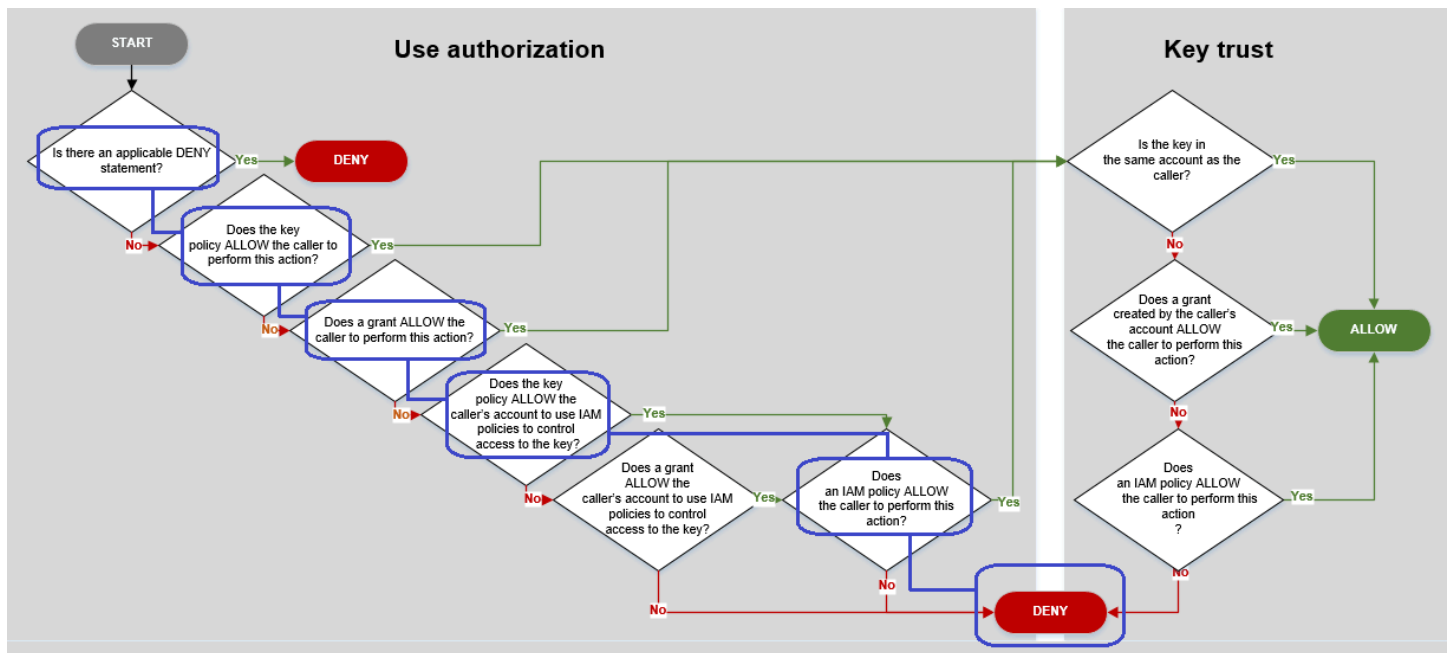
アクセス許可の例

- [例 1: ユーザーが の KMS キーへのアクセスを拒否される AWS アカウント](#)
- [例 2: ユーザーが別の で KMS キーを使用するアクセス許可を持つロールを引き受ける AWS アカウント](#)

例 1: ユーザーが の KMS キーへのアクセスを拒否される AWS アカウント

Alice は 111122223333 AWS アカウントの IAM ユーザーです。Alice は、同じ AWS アカウントの KMS キーへのアクセスを拒否されました。Alice が KMS キーを使用できないのはなぜでしょうか。

この場合、Alice が KMS キーへのアクセスを拒否されたのは、必要なアクセス許可を付与するキーポリシー、IAM ポリシー、権限がないためです。KMS キーのキーポリシーにより、AWS アカウント は IAM ポリシーを使用して KMS キーへのアクセスを制御できますが、どの IAM ポリシーも Alice に KMS キーを使用するアクセス許可を付与しません。



この例に関連するポリシーを考えてみます。

- Alice が使用する KMS キーには、[デフォルトキーポリシー](#)があります。このポリシーは、[KMS キーを所有する AWS アカウント](#) に、IAM ポリシーを使用して KMS キーへのアクセスを制御することを許可します。このキーポリシーは、フローチャートのキーポリシーが、発信者のアカウントが IAM ポリシーを使用してキーへのアクセスをコントロールすることを許可しているか?という条件を満たしています。

JSON

```
{
  "Version" : "2012-10-17",
  "Id" : "key-test-1",
  "Statement" : [ {
    "Sid" : "DelegateToIAMPolicies",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::111122223333:root"
    },
    "Action" : "kms:*",
    "Resource" : "*"
  } ]
}
```

- ただし、Alice に KMS キーの使用許可を付与するキーポリシー、IAM ポリシー、権限はありません。このため、Alice は KMS キーの使用許可を拒否されます。

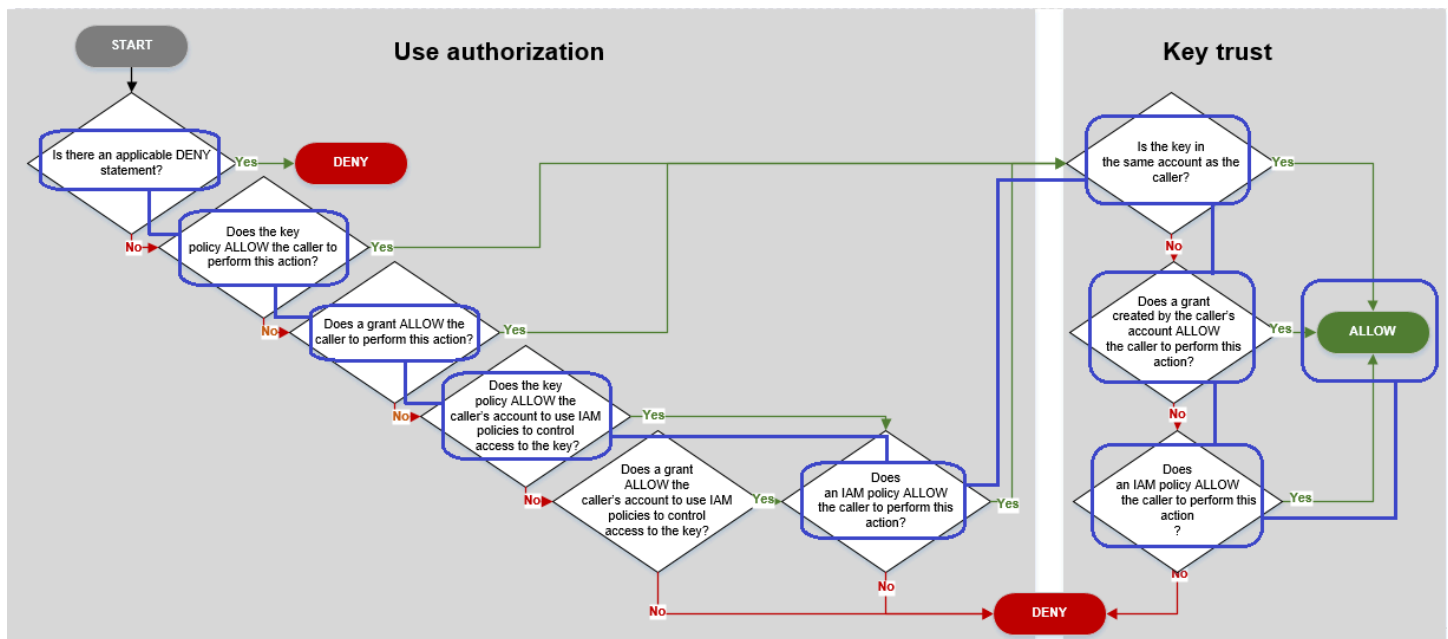
例 2: ユーザーが別の で KMS キーを使用するアクセス許可を持つロールを引き受ける AWS アカウント

Bob はアカウント 1 (111122223333) のユーザーです。彼は、[暗号化オペレーション](#)でアカウント 2 (444455556666) の KMS キーの使用を許可されています。これはどのようにすれば可能になるのでしょうか。

Tip

クロスアカウントのアクセス許可を評価するときは、キーポリシーが KMS キーのアカウントで指定されていることに注意してください。IAM ポリシーは、発信者が別のアカウントにいる場合でも、発信者のアカウントで指定されます。KMS キーへのクロスアカウントアクセス提供の詳細については、[他のアカウントのユーザーに KMS キーの使用を許可する](#) を参照してください。

- アカウント 2 の KMS キーのキーポリシーにより、アカウント 2 は IAM ポリシーを使用して KMS キーへのアクセスを制御できます。
- アカウント 2 の KMS キーのキーポリシーは、アカウント 1 が暗号化オペレーションで KMS キーを使用することを許可します。ただし、アカウント 1 は IAM ポリシーを使用して、プリンシパルに KMS キーへのアクセスを許可する必要があります。
- アカウント 1 の IAM ポリシーでは、Engineering ロールがアカウント 2 の KMS キーを暗号化オペレーションに使用することを許可します。
- アカウント 1 のユーザーである Bob には、Engineering ロールを引き受けるアクセス権限があります。
- Bob はこの KMS キーを信頼できます。この KMS キーは Bob のアカウントではありませんが、アカウントの IAM ポリシーにより、この KMS キーを使用する明示的なアクセス許可が Bob に付与されるためです。



アカウント 1 のユーザーである Bob が、アカウント 2 の KMS キーを使用することを許可するポリシーを考えてみます。

- KMS キーのキーポリシーにより、アカウント 2 (444455556666、KMS キーを所有するアカウント) は IAM ポリシーを使用して、KMS キーへのアクセスを制御できます。このキーポリシーでは、アカウント 1 (111122223333) に、KMS キーを暗号化オペレーション (ポリシーステートメントの Action 要素で指定) で使用することも許可します。ただし、プリンシパルに KMS キーへのアクセスを許可する IAM ポリシーがアカウント 1 で定義されるまでは、アカウント 1 のユーザーはアカウント 2 の KMS キーを使用できません。

フローチャートでは、アカウント 2 のこのキーポリシーは、キーポリシーは、呼び出し元のアカウントに IAM ポリシーを使用してキーへのアクセスを制御することを許可しますか? という条件を満たしています。

JSON

```
{
  "Id": "key-policy-acct-2",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PermissionUseIAMpolicies",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:root"
      }
    }
  ]
}
```

```

    },
    "Action": "kms:*",
    "Resource": "*"
  },
  {
    "Sid": "AllowAccount1UseThisKMSKeys",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:root"
    },
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncryptFrom",
      "kms:ReEncryptTo",
      "kms:GenerateDataKey",
      "kms:GenerateDataKeyWithoutPlaintext",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
]
}

```

- 呼び出し元 (AWS アカウント アカウント 1、111122223333) の IAM ポリシーは、アカウント 2 (444455556666) の KMS キーを使用して暗号化オペレーションを実行するアクセス許可をプリンシパルに付与します。Action 要素は、アカウント 2 のキーポリシーがアカウント 1 に付与したのと同じアクセス許可をプリンシパルに委任します。これらのアクセス許可をアカウント 1 の Engineering のロールに付与するために、[このインラインポリシーは Engineering のロールに埋め込まれています](#)。

このようなクロスアカウント IAM ポリシーは、アカウント 2 の KMS キーのキーポリシーが、KMS キーの使用許可をアカウント 1 に付与している場合にのみ有効です。また、アカウント 1 がプリンシパルに付与できるのは、キーポリシーがそのアカウントに付与しているアクションの実行アクセス許可のみです。

フローチャートでは、これは IAM ポリシーが呼び出し元にこのアクションを実行することを許可していますか? という条件を満たしています。

JSON

```
{
```

```

    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "kms:Encrypt",
          "kms:Decrypt",
          "kms:ReEncryptFrom",
          "kms:ReEncryptTo",
          "kms:GenerateDataKey",
          "kms:GenerateDataKeyWithoutPlaintext",
          "kms:DescribeKey"
        ],
        "Resource": [
          "arn:aws:kms:us-west-2:444455556666:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        ]
      }
    ]
  }
}

```

- 最後に必要な要素は、アカウント 1 での Engineering ロールの定義です。ロールの AssumeRolePolicyDocument により、Bob は Engineering ロールを引き受けることができます。

```

{
  "Role": {
    "Arn": "arn:aws:iam::111122223333:role/Engineering",
    "CreateDate": "2019-05-16T00:09:25Z",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": {
        "Principal": {
          "AWS": "arn:aws:iam::111122223333:user/bob"
        },
        "Effect": "Allow",
        "Action": "sts:AssumeRole"
      }
    },
    "Path": "/",
    "RoleName": "Engineering",
    "RoleId": "AR0A4KJY2TU23Y7NK62MV"
  }
}

```



```
}
```

AWS KMS アクセスコントロールの用語集

次のトピックでは、AWS KMS アクセスコントロールの重要な用語と概念について説明します。

認証

認証とは、アイデンティティを検証するプロセスです。リクエストを送信するには AWS KMS、AWS 認証情報 AWS を使用して にサインインする必要があります。

Authorization

認可は、AWS KMS リソースを作成、管理、または使用するためのリクエストを送信するアクセス許可を提供します。例えば、暗号化オペレーションで KMS キーを使用するには、権限が必要です。

AWS KMS リソースへのアクセスを制御するには、[キーポリシー](#)、[IAM ポリシー](#)、および[許可](#)を使用します。すべての KMS キーにはキーポリシーが必要です。キーポリシーで許可されている場合は、IAM ポリシーとgrantを使用して、プリンシパルに KMS キーへのアクセス権を付与することもできます。権限を絞り込むために、[条件キー](#)を使用して、リクエストまたはリソースが指定した条件を満たす場合に限り、アクセスを許可または拒否できます。また、[他の AWS アカウント](#) で、信頼するプリンシパルへのアクセスを許可できます。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けることによって、認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。AWS IAM Identity Center (IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して にアクセスすると、間接的 AWS にロールを引き受けます。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「AWS サインイン ユーザーガイド」の「[にサインインする方法 AWS アカウント](#)」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「[API リクエストに対するAWS Signature Version 4](#)」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、では、アカウントのセキュリティを強化するために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「[多要素認証](#)」および「IAM ユーザーガイド」の「[IAM のAWS 多要素認証](#)」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「[ルートユーザー認証情報が必要なタスク](#)」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーが、一時的な認証情報 AWS のサービス を使用して にアクセスするために ID プロバイダーとのフェデレーションを使用することを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Serviceアイデンティティセンターディレクトリのユーザー、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID が にアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「[What is IAM Identity Center?](#)」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

[IAM ユーザー](#)は、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウントを持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「[長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする](#)」を参照してください。

[IAM グループ](#)は、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「[IAM ユーザーに関するユースケース](#)」を参照してください。

IAM ロール

[IAM ロール](#)は、特定のアクセス許可 AWS アカウントを持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、[ユーザーから IAM ロールに切り替えることができます \(コンソール\)](#)。ロールを引き受けるには、または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「[ロールを引き受けるための各種方法](#)」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「[サードパーティー ID プロバイダー \(フェデレーション\) 用のロールを作成する](#)」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity

Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「[Permission sets](#)」を参照してください。

- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「[IAM でのクロスアカウントのリソースへのアクセス](#)」を参照してください。
- クロスサービスアクセス - 一部の では、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
- 転送アクセスセッション (FAS) - IAM ユーザーまたはロールを使用してアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービスへのリクエストのリクエストをリクエストする を組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「[転送アクセスセッション](#)」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける [IAM ロール](#)です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができます。詳細については、「IAM ユーザーガイド」の「[AWS のサービスに許可を委任するロールを作成する](#)」を参照してください。
- サービスにリンクされたロール - サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント、 サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、AWS CLI または AWS API リクエストを実行しているアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使えるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「[Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する](#)」を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御するには AWS、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは のオブジェクト AWS であり、アイデンティティまたはリソースに関連付けられると、そのアクセス許可を定義します。は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、IAM ユーザーガイドの [JSON ポリシー概要](#)を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースに必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、iam:GetRole アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、AWS Management Console、AWS CLI または AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「[カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する](#)」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「[管理ポリシーとインラインポリシーのいずれかを選択する](#)」を参照してください。

リソースベースのポリシー

AWS KMS [Ankey ポリシー](#)は、KMS キーへのアクセスを制御するリソースベースのポリシーです。すべての KMS キーにはキーポリシーが必要です。他の認可メカニズムを使用して、KMS キーへのアクセスを許可できますが、キーポリシーで許可されている場合に限り (キーポリシーで明示的に許可されていなくても、IAM ポリシーを使用して KMS キーへのアクセスを「拒否」できます)。

リソースベースのポリシーとは、JSON ポリシードキュメントです。KMS キーなどのリソースにアタッチして、特定のリソースへのアクセスを制御します。リソースベースのポリシーは、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件を定義します。リソースベースのポリシーではリソースを指定しませんが、アカウント、ユーザー、ロール、フェデレーティッドユーザー、などのプリンシパルを指定する必要があります AWS のサービス。リソースベースのポリシーは、リソースを管理するそのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、AWS 管理ポリシーなど、IAM の [AWSKeyManagementServicePowerUser 管理ポリシー](#)を使用できません。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- **アクセス許可の境界** - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。Principal フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「[IAM エンティティのアクセス許可の境界](#)」を参照してください。
- **サービスコントロールポリシー (SCPs)** – SCPsは、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。AWS Organizations は、ビジネスが所

有する複数の AWS アカウント をグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「[サービスコントロールポリシー \(SCP\)](#)」を参照してください。

- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations 「ユーザーガイド」の AWS のサービス「[リソースコントロールポリシー \(RCPs\)](#)」を参照してください。RCPs
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「[セッションポリシー](#)」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関係する場合に がリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「[ポリシー評価ロジック](#)」を参照してください。

AWS KMS リソース

では AWS KMS、プライマリリソースは です AWS KMS key。 は、KMS キーにわかりやすい名前を提供する独立したリソースであるエイリアス [AWS KMS](#) もサポートしています。一部の AWS KMS オペレーションでは、エイリアスを使用して KMS キーを識別できます。

KMS キーまたはエイリアスの各インスタンスには、標準形式の一意の [Amazon リソースネーム](#) (ARN) があります。AWS KMS リソースでは、AWS サービス名は です kms。

- AWS KMS key

ARN 形式:

arn:*AWS partition name*:*AWS service name*:*AWS #####*:*AWS ##### ID*:key/*key ID*

ARN の例:

arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab

- エイリアス

ARN 形式:

arn:*AWS partition name*:*AWS service name*:*AWS #####*:*AWS ##### ID*:alias/*alias name*

ARN の例:

arn:aws:kms:us-west-2:111122223333:alias/example-alias

AWS KMS には、AWS KMS リソースを操作するための一連の API オペレーションが用意されています。AWS Management Console および AWS KMS API オペレーションでの KMS キーの識別の詳細については、「」を参照してください[キー識別子 \(KeyId\)](#)。AWS KMS オペレーションのリストについては、[AWS Key Management Service API リファレンス](#)を参照してください。

KMS キーを作成する

は、AWS KMS keys で作成することも AWS Management Console、[CreateKey](#) オペレーションまたは [AWS::KMS::Key AWS CloudFormation リソース](#) を使用して作成することもできます。このプロセスでは、KMS キーのキーポリシーも設定します。これはいつでも変更できます。また、作成する KMS キーのタイプを定義する以下の値も選択します。KMS キー作成後にこれらのプロパティを変更することはできません。

KMS キータイプ

キータイプは、作成される暗号化キーのタイプを決定するプロパティです。は、データを保護するために 3 つのキータイプ AWS KMS を提供します。

- Advanced Encryption Standard (AES) 対称キー

AES の Galois Counter Mode (GCM) モードで使用される 256 ビットキーで、4KB 未満のサイズのデータの認証付き暗号化/復号化を提供します。これは最も一般的なタイプのキーであり、ユーザーに代わってデータを暗号化 AWS のサービス するアプリケーションおよびで使用される他のデータ暗号化キーを保護するために使用されます。

- RSA、楕円曲線、または SM2 (中国リージョンのみ) の非対称キー

これらのキーはさまざまなサイズで利用可能で、多くのアルゴリズムをサポートしています。選択したアルゴリズムに応じて、暗号化と復号化、署名と検証、もしくは共有シークレットの取得のいずれかのオペレーションに使用できます。

- Hash-based Message Authentication Code (HMAC) オペレーションを実行するための対称キー

これらのキーは、署名および検証オペレーションに使用される 256 ビットのキーです。

KMS キーは、サービスからプレーンテキストでエクスポートすることはできません。これらはサービスに使用されるハードウェアセキュリティモジュール (HSM) により生成され、そのモジュール内でのみ使用できます。これは、キーが侵害されないように AWS KMS するための基本的なセキュリティプロパティです。

キーの用途

キー用途は、キーがサポートする暗号化オペレーションを決定するプロパティです。KMS キーには、ENCRYPT_DECRYPT、SIGN_VERIFY、GENERATE_VERIFY_MAC、KEY_AGREEMENT のいずれかのキー用途を指定できます。各 KMS キーに指定できるキー用途は 1 つだけです。これは、[米国国立標準技術研究所 \(NIST\) の「Special Publication 800-57 Recommendations for Key](#)

[Management](#)」のセクション 5.2 「Key usage」に従った主要な使用のベストプラクティスに従います。KMS キーを複数タイプのオペレーションに使用すると、両方のオペレーションの成果が攻撃に対してより脆弱になります。

キー仕様

キー仕様は、キーの暗号化設定を表すプロパティです。キー仕様の意味は、キータイプによって異なります。

KMS キーにおいて、キー仕様はその KMS キーが対称か非対称かを決定します。また、そのキーマテリアルのタイプとサポートするアルゴリズムも決定します。

デフォルトのキースペックである [SYMMETRIC_DEFAULT](#) は、256 ビット対称暗号化キーを表します。サポートされているすべてのキー仕様の詳細については、「[キー仕様のリファレンス](#)」を参照してください。

キーマテリアルのオリジン

キーマテリアルのオリジンは、KMS キーのキーマテリアルのソースを識別する KMS キープロパティです。KMS キーの作成時にキーマテリアルのオリジンを選択し、それを変更することはできません。キーマテリアルのソースは、KMS キーのセキュリティ、耐久性、可用性、レイテンシー、スループット特性に影響します。

各 KMS キーには、メタデータにキーマテリアルへの参照が含まれています。対称暗号化 KMS キーのキーマテリアルのオリジンは、異なる可能性があります。AWS KMS が生成するキーマテリアル、または[カスタムキーストア](#)内で生成されるキーマテリアルを使用するか、もしくは[独自のキーマテリアルをインポートすることができます](#)。

デフォルトでは、各 KMS キーに一意のキーマテリアルがあります。ただし、同じキーマテリアルで[マルチリージョンキー](#)のセットを作成することができます。

KMS キーには、EXTERNAL ([インポートされたキーマテリアル](#)) AWS_KMS、AWS_CLOUDHSM (キー[ストアの KMS キー](#))、または ([外部 AWS CloudHSM](#)キーストアの KMS キー) のいずれかのキーマテリアルオリジン値を指定できます。EXTERNAL_KEY_STORE [???](#)

トピック

- [KMS キーを作成するためのアクセス許可](#)
- [作成する KMS キーのタイプの選択](#)
- [対称暗号化 KMS キーを作成する](#)

- [非対称 KMS キーを作成する](#)
- [HMAC KMS キーを作成する](#)
- [マルチリージョンのプライマリキーを作成する](#)
- [マルチリージョンのレプリカキーを作成する](#)
- [インポートされたキーマテリアルを持つ KMS キーの作成](#)
- [キーストアに KMS AWS CloudHSM キーを作成する](#)
- [外部キーストアで KMS キーを作成する](#)

KMS キーを作成するためのアクセス許可

コンソールで、または API を使用して KMS キーを作成するには、IAM ポリシーで次のアクセス許可を持っている必要があります。可能な限り、[条件キー](#)を使用してアクセス許可を制限します。例えば、IAM ポリシーで [kms:KeySpec](#) 条件キーを使用して、プリンシパルが対称暗号化キーのみを作成できるようにすることが可能です。

キーを作成するプリンシパルの IAM ポリシーの例については、「[KMS キーの作成をユーザーに許可する](#)」を参照してください。

Note

タグとエイリアスを管理する許可をプリンシパルに付与する場合は注意が必要です。タグまたはエイリアスを変更すると、カスターマネージドキーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

- [kms:CreateKey](#) は必須です。
- [kms:CreateAlias](#) は、新しい KMS キーごとにエイリアスが必要なコンソールで、KMS キーを作成するために必要です。
- [kms:TagResource](#) は、KMS キーの作成中にタグを追加するために必要です。
- [iam:CreateServiceLinkedRole](#) は、マルチリージョンのプライマリキーを作成するために必要です。詳細については、「[マルチリージョンキーへのアクセスを制御する](#)」を参照してください。

[kms:PutKeyPolicy](#) アクセス許可は、KMS キーの作成には必要ありません。kms:CreateKey アクセス許可には、初期キーポリシーを設定する許可が含まれています。ただし、KMS キーの作成時にこ

のアクセス許可をキーポリシーに追加して、KMS キーへのアクセスを制御できるようにする必要があります。代替方法として、[BypassLockoutSafetyCheck](#) パラメータを使用します。これは推奨されていません。

KMS キーは、作成された AWS アカウントに属します。KMS キーを作成する IAM ユーザーはキー所有者とはみなされないため、作成した KMS キーを使用または管理するためのアクセス許可が自動的に付与されません。他のプリンシパルと同様に、キー作成者は、キーポリシー、IAM ポリシー、またはグラントを使用してアクセス許可を取得する必要があります。ただし、`kms:CreateKey` アクセス許可を持つプリンシパルは、初期キーポリシーを設定し、キーを使用または管理するためのアクセス許可を自分自身に付与できます。

作成する KMS キーのタイプの選択

作成する KMS キーのタイプは、予定される KMS キーの用途、セキュリティ要件、および認可要件に大きく依存します。KMS キーのキータイプとキー用途によって、そのキーで実行できる暗号化オペレーションが決まります。キーの用途は、各 KMS キーに 1 つしかありません。KMS キーを複数タイプのオペレーションに使用すると、すべてのオペレーションの成果が攻撃に対してより脆弱になります。

プリンシパルが特定のキーの用途だけに KMS キーを作成できるようにするには、[kms:KeyUsage](#) 条件キーを使用します。kms:KeyUsage 条件キーを使用して、プリンシパルがキーの用途に基づいて KMS キーの API オペレーションを呼び出せるようにすることもできます。例えば、キーの用途が SIGN_VERIFY である場合にのみ、KMS キーを無効にするアクセス許可を許可できます。

以下のガイダンスを使用し、ユースケースに基づいて必要なタイプの KMS キーを決定します。

データの暗号化と復号

データの暗号化と復号を必要とするほとんどのユースケースには、[対称 KMS キー](#)を使用します。AWS KMS が使用する対称暗号化アルゴリズムは、高速かつ効率的で、データの機密性と信頼性を保証します。これは、[暗号化コンテキスト](#)として定義された追加認証データ (AAD) による認証された暗号化をサポートします。このタイプの KMS キーでは、暗号化されたデータの送信者と受信者の両方が呼び出す有効な AWS 認証情報を持っている必要があります AWS KMS。

を呼び出すことができないユーザーが の外部 AWS で暗号化を必要とするユースケースでは AWS KMS、[非対称 KMS キー](#)が適しています。非対称 KMS キーのパブリックキーを配信して、これらのユーザーがデータを暗号化できるようにします。また、そのデータを復号する必要があるアプリケーションは、AWS KMS内で非対称 KMS キーのプライベートキーを使用できます。

メッセージの署名および署名の検証

メッセージに署名して、署名を検証するには、[非対称 KMS キー](#)を使用する必要があります。KMS キーは、RSA キーペア、楕円曲線 (ECC) キーペア、ML-DSA キーペア、または SM2 キーペアを表すキー[仕様](#)で使用できます (中国リージョンのみ)。選択するキー仕様は、使用する署名アルゴリズムによって決まります。ECC キーペアがサポートする ECDSA 署名アルゴリズムは、RSA 署名アルゴリズムよりも推奨されます。RSA または ECC キーからポスト量子キーに移行するときは、ML-DSA キーペアを使用します。ただし、AWS 外部で署名を検証するユーザーをサポートするには、特定のキー仕様と署名アルゴリズムを使用することが必要になる場合があります。

非対称キーペアを用いた暗号化

非対称キーペアを用いてデータを暗号化するには、[RSA キー仕様](#)または [SM2 キー仕様](#) (中国リージョンのみ) の[非対称 KMS キー](#)を使用する必要があります。KMS キーペアのパブリックキーを使用して、AWS KMS でデータを暗号化するには、[Encrypt](#) オペレーションを使用します。[パブリックキーをダウンロード](#)して、外部でデータを暗号化する必要がある当事者と共有することもできます AWS KMS。

非対称 KMS キーのパブリックキーをダウンロードする場合、AWS KMS の外部で使用できます。ただし、KMS キーを保護するセキュリティコントロールの対象ではなくなりました AWS KMS。たとえば、パブリック AWS KMS キーの使用を制御するためにキーポリシーや許可を使用することはできません。また、AWS KMS がサポートする暗号化アルゴリズムを使用して、キーが暗号化と復号にのみ使用されるかどうかを制御することもできません。詳細については、「[パブリックキーのダウンロードに関する特別な考慮事項](#)」を参照してください。

の外部でパブリックキーで暗号化されたデータを復号するには AWS KMS、[Decrypt](#) オペレーションを呼び出します。キー用途を SIGN_VERIFY に設定して KMS キーからのパブリックキーでデータを暗号化した場合、Decrypt オペレーションは失敗します。また、選択したキー仕様をサポート AWS KMS していないアルゴリズムを使用して暗号化された場合も失敗します。主なキー仕様とサポートされているアルゴリズムの詳細については、「[キー仕様のリファレンス](#)」を参照してください。

これらのエラーを回避するには、の外部でパブリックキーを使用するすべてのユーザーがキー設定を保存 AWS KMS する必要があります。AWS KMS コンソールと [GetPublicKey](#) レスポンスは、パブリックキーを共有するときに含める必要がある情報を提供します。

共有シークレットを取得する

共有シークレットを取得するには、[NIST が推奨する楕円曲線](#)または [SM2](#) (中国リージョンのみ) キーマテリアルを持つ KMS キーを使用します。は、[楕円曲線のパブリック/プライバー](#)

[トキープアから共有シークレットを取得することで、楕円曲線暗号コファクタ Diffie-Hellman Primitive](#) (ECDH) AWS KMS を使用して 2 つのピア間でキーアグリーメントを確立します。[DeriveSharedSecret](#) オペレーションが返す raw 共有シークレットを使用して、2 つの当事者間で送信されるデータを暗号化および復号できる対称キーを導き出すか、HMACs を生成して検証できます。raw 共有シークレットを使用して対称キーを導き出すときは、[キー導き出しに関する NIST の推奨事項](#)に従う AWS KMS ことをお勧めします。

HMAC コードを生成して検証する

Hash-based Message Authentication Code を生成して検証するには、HMAC KMS キーを使用します。で HMAC キーを作成すると AWS KMS、はキーマテリアル AWS KMS を作成および保護し、キーに正しい MAC アルゴリズムを使用するようにします。HMAC コードは、擬似乱数としての使用、および対称署名とトークン化のための特定のシナリオでの使用も可能です。

HMAC KMS キーは対称キーです。AWS KMS コンソールで HMAC KMS キーを作成するときは、Symmetric キータイプを選択します。

AWS サービスで を使用する

[AWS と統合された サービス AWS KMS](#)で使用する KMS キーを作成するには、データを暗号化する service. AWS services のドキュメントで[対称暗号化 KMS キー](#)が必要です。

これらの考慮事項に加えて、キー仕様が異なる KMS キーの暗号化オペレーションでは、料金とリクエストクォータも異なります。AWS KMS の料金については、[AWS Key Management Service の料金](#)を参照してください。リクエストクォータの詳細については、「[クォータのリクエスト](#)」を参照してください。

対称暗号化 KMS キーを作成する

このトピックでは、基本的な KMS キーを作成する方法について説明します。これは、キーマテリアルを含む 1 つのリージョンの[対称暗号化 KMS キー](#)です AWS KMS。この KMS キーを使用して、AWS のサービス内のリソースを保護することができます。

対称暗号化 KMS キーは、AWS KMS コンソール、[CreateKey](#) API、または [AWS::KMS::Key AWS CloudFormation テンプレート](#)を使用して作成できます。

デフォルトのキー仕様である [SYMMETRIC_DEFAULT](#) は、対称暗号化 KMS キー向けキー仕様です。AWS KMS コンソールで対称キータイプと暗号化および復号キーの使用を選択すると、SYMMETRIC_DEFAULTキー仕様が選択されます。[CreateKey](#) オペレーションで、KeySpec 値を

指定しない場合、SYMMETRIC_DEFAULT が選択されます。別のキー仕様を使用する理由がない場合は、SYMMETRIC_DEFAULT を選択することをお勧めします。

KMS キーに適用されるクォータの詳細については、[クォータ](#) を参照してください。

AWS KMS コンソールの使用

を使用して AWS KMS keys (KMS キー) AWS Management Console を作成できます。

Important

エイリアス、説明、またはタグには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。
5. 対称暗号化 KMS キーを作成するには、[Key type] (キーのタイプ) で [Symmetric] (対称) を選択します。
6. [Key usage] (キーの使用) では、[Encrypt and decrypt] (暗号化および復号) オプションがすでに選択されています。
7. [次へ] を選択します。
8. KMS キーのエイリアスを入力します。エイリアス名の先頭を **aws/** にすることはできません。**aws/** プレフィックスは、アカウント AWS マネージドキー で表すために Amazon Web Services によって予約されています。

Note

エイリアスを追加、削除、更新すると、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[エイリアスを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

エイリアスは KMS キーを識別するために使用する表示名です。保護する予定のデータタイプ、または KMS キーで使用する予定のアプリケーションを示すエイリアスを選択することをお勧めします。

エイリアスは AWS Management Console で KMS キーを作成するときに必要です。 [CreateKey](#) オペレーションを使用する場合、これらのオペレーションはオプションです。

9. (オプション) KMS キーの説明を入力します。

今すぐ説明を追加するか、[キーの状態](#)が Pending Deletion または Pending Replica Deletion でない限り、後でいつでも更新できます。既存のカスタマーマネージドキーの説明を追加、変更、または削除するには、の KMS キーの詳細ページの説明を編集する AWS Management Console か、 [UpdateKeyDescription](#) オペレーションを使用します。

10. (オプション) タグキーとオプションのタグ値を入力します。KMS キーに複数のタグを追加するには、[Add tag] (タグを追加) を選択します。

 Note

KMS キーのタグ付けまたはタグ解除により、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[タグを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、[のタグ AWS KMS](#) および [の ABAC AWS KMS](#) を参照してください。

11. [次へ] を選択します。

12. KMS キーを管理できる IAM ユーザーとロールを選択します。

 メモ

このキーポリシーは、この KMS キー AWS アカウント を完全に制御します。これにより、アカウント管理者は IAM ポリシーを使用して、他のプリンシパルに KMS キーを管理する許可を付与できます。詳細については、「[the section called “デフォルトのキーポリシー”](#)」を参照してください。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子のキーポリシーにキー管理者を追加します "Allow access for Key Administrators"。このステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

13. (オプション) 選択した IAM ユーザーとロールがこの KMS キーを削除しないようにするには、ページの下部にある [Key deletion] (キーの削除) セクションで、[Allow key administrators to delete this key] (キー管理者にこのキーの削除を許可する) のチェックボックスをオフにします。
14. [次へ] を選択します。
15. [暗号化オペレーション](#) でキーを使用できる IAM ユーザーとロールを選択します。

メモ

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子 "Allow use of the key" およびのキーポリシーにキーユーザーを追加します "Allow attachment of persistent resources"。これらのステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

16. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある Other AWS アカウント セクションで、Add another AWS アカウントを選択し、外部アカウントの AWS アカウント 識別番号を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

Note

外部アカウントでプリンシパルが KMS キーを使用できるようにするには、外部アカウントの管理者が、これらのアクセス許可を付与する IAM ポリシーを作成する必要があります。

ます。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

17. [Next] (次へ) を選択します。
18. キーのキーポリシーステートメントを確認します。キーポリシーを変更するには、**編集** を選択します。
19. [次へ] を選択します。
20. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。
21. [Finish] (完了) を選択し、KMS キーを作成します。

AWS KMS API の使用

[CreateKey](#) オペレーションを使用して、すべてのタイプの AWS KMS keys を作成できます。これらの例では、[AWS Command Line Interface \(AWS CLI\)](#) を使用します。複数のプログラミング言語の例については、「[AWS SDK または CLI CreateKey で使用する](#)」を参照してください。

Important

Description フィールドまたは Tags フィールドには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

次のオペレーションでは、AWS KMSで生成されるキーマテリアルにバックアップされた、単一リージョン内の対称暗号化キーを作成します。このオペレーションには必須パラメータはありません。ただし、キーポリシーを指定するために Policy パラメータが必要になる場合もあります。キーポリシー ([PutKeyPolicy](#)) を変更したり、[説明](#) や [タグ](#) などのオプション要素をいつでも追加したりできます。また、[非対称キー](#)、[マルチリージョンキー](#)、[インポートされたキーマテリアル](#)を含むキー、および[カスタムキーストア](#)内のキーも作成できます。クライアント側暗号化用のデータキーを作成するには、[GenerateDataKey](#) オペレーションを使用します。

CreateKey オペレーションでは、エイリアスを指定することはできませんが、[CreateAlias](#) オペレーションを実行して、新しい KMS キーのエイリアスを作成できます。

次の例では、パラメータを指定せずに CreateKey オペレーションを呼び出します。このコマンドでは、すべてのデフォルト値が使用されます。これは、AWS KMSが生成したキーマテリアルで対称暗号化 KMS キーを作成します。

```
$ aws kms create-key
{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "Enabled": true,
    "KeySpec": "SYMMETRIC_DEFAULT",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "CreationDate": 1502910355.475,
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "MultiRegion": false
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
  }
}
```

新規の KMS キーにキーポリシーを指定しない場合、CreateKey が適用する [デフォルトのキーポリシー](#) は、新規の KMS キーを作成するときに使用され、コンソールが適用するデフォルトのキーポリシーとは異なります。

たとえば、[GetKeyPolicy](#) オペレーションに対するこの呼び出しは、CreateKey 適用されるキーポリシーを返します。これにより、KMS キー AWS アカウント へのアクセスが許可され、KMS キーの AWS Identity and Access Management (IAM) ポリシーを作成できます。IAM ポリシーおよび KMS キーのキーポリシーの詳細については、「[KMS キーのアクセスとアクセス許可](#)」を参照してください。

```
$ aws kms get-key-policy --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --policy-name
default --output text
```

JSON

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

```
"Id" : "key-default-1",
"Statement" : [ {
  "Sid" : "EnableIAMUserPermissions",
  "Effect" : "Allow",
  "Principal" : {
    "AWS" : "arn:aws:iam::111122223333:root"
  },
  "Action" : "kms:*",
  "Resource" : "*"
} ]
}
```

非対称 KMS キーを作成する

AWS KMS コンソール操作で、または [CreateKey](#) API もしくは [AWS::KMS::Key](#) [AWS CloudFormation テンプレート](#) を使用して、[非対称 KMS キー](#) を作成できます。非対称 KMS キーは、暗号化、署名、または共有シークレットの取得に使用する公開キーとプライベートキーのキーペアを表します。プライベートキーは内に残ります AWS KMS。の外部で使用するパブリックキーをダウンロードするには AWS KMS、「」を参照してください [パブリックキーをダウンロードする](#)。

非対称 KMS キーを作成するときは、キー仕様を選択する必要があります。選択するキー仕様は、多くの場合、規制、セキュリティ、ビジネス要件によって決定されます。また、暗号化または署名する必要があるメッセージのサイズによっても影響を受ける可能性があります。一般に、長い暗号化キーは、ブルートフォース攻撃に対してより耐性があります。サポートされているすべてのキー仕様の詳細については、「[キー仕様のリファレンス](#)」を参照してください。

AWS と統合する サービスは、非対称 KMS キーをサポート AWS KMS していません。AWS サービスに保存または管理するデータを暗号化する KMS キーを作成する場合は、[対称暗号化 KMS キーを作成します](#)。

KMS キーの作成に必要なアクセス許可については、[KMS キーを作成するためのアクセス許可](#) を参照してください。

AWS KMS コンソールの使用

を使用して非対称 AWS KMS keys (KMS キー) AWS Management Console を作成できます。各非対称 KMS キーは、公開キーとプライベートキーのキーペアを表します。

⚠ Important

エイリアス、説明、またはタグには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。
5. 非対称 KMS キーを作成するには、[Key type] (キータイプ) で [Asymmetric] (非対称) を選択します。
6. 公開キー暗号化用の非対称 KMS キーを作成するには、[キーの使用] (キーの使用方法) で [Encrypt and decrypt] (暗号化と復号) を選択します。

メッセージに署名して署名を検証するための非対称 KMS キーを作成するには、[キーの使用] (キーの使用方法) で [署名と検証] を選択します。

共有シークレットを取得するための非対称 KMS キーを作成するには、[キーの使用] で [キーアグリーメント] を選択します。

キー使用法の値の選択については、を参照してください [作成する KMS キーのタイプの選択](#)。

7. 非対称 KMS キーの仕様 ([Key spec] (キー仕様)) を選択します。
8. [次へ] を選択します。
9. KMS キーの [エイリアス](#) を入力します。エイリアス名の先頭を **aws/** にすることはできません。この **aws/** プレフィックスは、アカウント内の AWS マネージドキーを表すために、Amazon Web Services によって予約されます。

エイリアスは、コンソールおよび一部の AWS KMS APIs。保護する予定のデータタイプ、または KMS キーで使用する予定のアプリケーションを示すエイリアスを選択することをお勧めします。

エイリアスは AWS Management Console で KMS キーを作成するときに必要です。 [CreateKey](#) オペレーションの使用時にエイリアスを指定することはできません。ただし、コンソールまたは

[CreateAlias](#) オペレーションを使用して、既存の KMS キーのエイリアスを作成できます。詳細については、「[のエイリアス AWS KMS](#)」を参照してください。

10. (オプション) KMS キーの説明を入力します。

保護する予定のデータタイプ、または KMS キーで使用する予定のアプリケーションを表す説明を入力します。

今すぐ説明を追加するか、[キーの状態](#)が Pending Deletion または Pending Replica Deletion でない限り、後でいつでも更新できます。既存のカスタマーマネージドキーの説明を追加、変更、または削除するには、の KMS キーの詳細ページの説明を編集する AWS Management Console か、[UpdateKeyDescription](#) オペレーションを使用します。

11. (オプション) タグキーとオプションのタグ値を入力します。KMS キーに複数のタグを追加するには、[Add tag] (タグを追加) を選択します。

AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、[のタグ AWS KMS](#) および [の ABAC AWS KMS](#) を参照してください。

12. [次へ] を選択します。

13. KMS キーを管理できる IAM ユーザーとロールを選択します。

 メモ

このキーポリシーは、この KMS キー AWS アカウント を完全に制御します。これにより、アカウント管理者は IAM ポリシーを使用して、他のプリンシパルに KMS キーを管理する許可を付与できます。詳細については、「[the section called “デフォルトのキーポリシー”](#)」を参照してください。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子のキーポリシーにキー管理者を追加します "Allow access for Key Administrators"。このステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

14. (オプション) 選択した IAM ユーザーとロールがこの KMS キーを削除しないようにするには、ページの下部にある [Key deletion] (キーの削除) セクションで、[Allow key administrators to delete this key] (キー管理者にこのキーの削除を許可する) のチェックボックスをオフにします。
15. [次へ] を選択します。
16. [暗号化オペレーション](#)で KMS キーを使用できる IAM ユーザーとロールを選択します。

メモ

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子 "Allow use of the key" およびのキーポリシーにキーユーザーを追加します "Allow attachment of persistent resources"。これらのステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

17. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある [Other AWS アカウント] セクションで、[Add another AWS アカウント] を選択し、外部アカウントの AWS アカウント ID 番号を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

Note

外部アカウントでプリンシパルが KMS キーを使用できるようにするには、外部アカウントの管理者が、これらの許可を付与する IAM ポリシーを作成する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

18. [Next] (次へ) を選択します。
19. キーのキーポリシーステートメントを確認します。キーポリシーを変更するには、編集を選択します。
20. [次へ] を選択します。
21. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。
22. [Finish] (完了) を選択し、KMS キーを作成します。

AWS KMS API の使用

[CreateKey](#) オペレーションを使用して、非対称を作成できます AWS KMS key。以下の例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

非対称 KMS キーを作成する場合は、作成するキーのタイプを決定する KeySpec パラメータを指定する必要があります。また、ENCRYPT_DECRYPT、SIGN_VERIFY、KEY_AGREEMENT の KeyUsage 値も指定する必要があります。KMS キー作成後にこれらのプロパティを変更することはできません。

CreateKey オペレーションでは、エイリアスを指定することはできませんが、[CreateAlias](#) オペレーションを実行して、新しい KMS キーのエイリアスを作成できます。

Important

Description フィールドまたは Tags フィールドには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

公開暗号化用の非対称 KMS キーペアを作成する

次の例では、CreateKey オペレーションを使用して、公開キーの暗号化用に設計された 4096 ビット RSA キーの非対称 KMS キーを作成します。

```
$ aws kms create-key --key-spec RSA_4096 --key-usage ENCRYPT_DECRYPT
{
  "KeyMetadata": {
    "KeyState": "Enabled",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "Description": "",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": 1569973196.214,
    "MultiRegion": false,
    "KeySpec": "RSA_4096",
    "CustomerMasterKeySpec": "RSA_4096",
    "KeyUsage": "ENCRYPT_DECRYPT",
```



```

    "EncryptionAlgorithms": [
      "RSAES_OAEP_SHA_1",
      "RSAES_OAEP_SHA_256"
    ],
    "AWSAccountId": "111122223333",
    "Origin": "AWS_KMS",
    "Enabled": true
  }
}

```

署名および検証用の非対称 KMS キーペアを作成する

次のコマンド例では、署名と検証に使用する ECC キーのペアを表す非対称 KMS キーを作成します。暗号化と復号のために楕円曲線キーペアを作成することはできません。

```

$ aws kms create-key --key-spec ECC_NIST_P521 --key-usage SIGN_VERIFY
{
  "KeyMetadata": {
    "KeyState": "Enabled",
    "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "CreationDate": 1570824817.837,
    "Origin": "AWS_KMS",
    "SigningAlgorithms": [
      "ECDSA_SHA_512"
    ],
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
    "AWSAccountId": "111122223333",
    "KeySpec": "ECC_NIST_P521",
    "CustomerMasterKeySpec": "ECC_NIST_P521",
    "KeyManager": "CUSTOMER",
    "Description": "",
    "Enabled": true,
    "MultiRegion": false,
    "KeyUsage": "SIGN_VERIFY"
  }
}

```

共有シークレットを取得するための非対称 KMS キーペアを作成する

次のコマンド例では、共有シークレットの取得に使用する ECDH キーのペアを表す非対称 KMS キーを作成します。暗号化と復号のために楕円曲線キーペアを作成することはできません。

```
$ aws kms create-key --key-spec ECC_NIST_P256 --key-usage KEY_AGREEMENT
{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
    "CreationDate": "2023-12-27T19:10:15.063000+00:00",
    "Enabled": true,
    "Description": "",
    "KeyUsage": "KEY_AGREEMENT",
    "KeyState": "Enabled",
    "Origin": "AWS_KMS",
    "KeyManager": "CUSTOMER",
    "CustomerMasterKeySpec": "ECC_NIST_P256",
    "KeySpec": "ECC_NIST_P256",
    "KeyAgreementAlgorithms": [
      "ECDH"
    ],
    "MultiRegion": false
  }
}
```

HMAC KMS キーを作成する

HMAC KMS キーは、AWS KMS コンソール、[CreateKey](#) API、または [AWS::KMS::Key AWS CloudFormation テンプレート](#) を使用して作成できます。

HMAC KMS キーを作成するときは、キー仕様を選択する必要があります。は、[HMAC KMS キーの複数のキー仕様](#) AWS KMS をサポートしています。ユーザーが選択するキー仕様は、規制、セキュリティ、またはビジネス要件に応じて決定される場合があります。一般に、長いキーはブルートフォース攻撃に対する耐性が高くなります。

KMS キーの作成に必要なアクセス許可については、[KMS キーを作成するためのアクセス許可](#) を参照してください。

AWS KMS コンソールの使用

を使用して HMAC KMS キー AWS Management Console を作成できます。HMAC KMS キーは、キーの用途が [Generate and verify MAC] (MAC の生成と検証) である対称キーです。マルチリージョンの HMAC キーを作成することもできます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。
5. [キーの種類] で、[対称] を選択します。

HMAC KMS キーは対称です。同じキーを使用して、HMAC タグの生成と検証を行います。

6. [Key usage] (キーの使用) には、[Generate and verify MAC] (MAC の生成と検証) を選択します。

MAC の生成と検証は、HMAC KMS キーに対して唯一有効なキーの用途です。

 Note

対称キーに対する [Key usage] (キーの使用) は、選択されたリージョンで HMAC KMS キーがサポートされている場合にのみ表示されます。

7. HMAC KMS キーの仕様 ([Key spec] (キーの仕様)) を選択します。

選択するキーの仕様は、規制、セキュリティ、またはビジネス要件に応じて決定できます。一般に、キーが長いほど安全性が高くなります。

8. [マルチリージョン](#) のプライマリ HMAC キーを作成するには、[Advanced options] (詳細オプション) で [Multi-Region key] (マルチリージョンキー) を選択します。この KMS キーに定義する [共有プロパティ](#) (キーのタイプとキーの用途など) は、そのレプリカキーと共有されます。

この手順を使用してレプリカキーを作成することはできません。マルチリージョンのレプリカ HMAC キーを作成するには、[レプリカキーを作成するための手順](#)に従ってください。

9. [次へ] を選択します。
10. KMS キーの [エイリアス](#) を入力します。エイリアス名の先頭を **aws/** にすることはできません。この **aws/** プレフィックスは、アカウント内の AWS マネージドキーを表すために、Amazon Web Services によって予約されます。

KMS キーを HMAC キーとして識別するエイリアス (HMAC/test-key など) の使用をお勧めします。これにより、AWS KMS タグやエイリアスでキーをソートおよびフィルタリングできるコンソールで HMAC キーを簡単に識別できますが、キー仕様やキーの使用法では識別できません。

エイリアスは AWS Management Console で KMS キーを作成するときに必要です。[CreateKey](#) オペレーションの使用時にエイリアスを指定することはできません。ただし、コンソールまたは [CreateAlias](#) オペレーションを使用して、既存の KMS キーのエイリアスを作成できます。詳細については、「[のエイリアス AWS KMS](#)」を参照してください。

11. (オプション) KMS キーの説明を入力します。

保護する予定のデータタイプ、または KMS キーで使用する予定のアプリケーションを表す説明を入力します。

今すぐ説明を追加するか、[キーの状態](#)が Pending Deletion または Pending Replica Deletion でない限り、後でいつでも更新できます。既存のカスタマーマネージドキーの説明を追加、変更、または削除するには、[の](#)で KMS キーの詳細ページの説明を編集する AWS Management Console AWS Management Console が、[UpdateKeyDescription](#) オペレーションを使用します。

12. (オプション) タグキーとオプションのタグ値を入力します。KMS キーに複数のタグを追加するには、[Add tag] (タグを追加) を選択します。

Type=HMAC など、キーを HMAC キーとして識別するタグの追加を検討してください。これにより、AWS KMS タグやエイリアスでキーをソートおよびフィルタリングできるコンソールで HMAC キーを簡単に識別できますが、キー仕様やキーの使用法では識別できません。

AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、[のタグ AWS KMS](#) および [の ABAC AWS KMS](#) を参照してください。

13. [次へ] を選択します。

14. KMS キーを管理できる IAM ユーザーとロールを選択します。

 メモ

このキーポリシーは、この KMS キー AWS アカウント を完全に制御します。これにより、アカウント管理者は IAM ポリシーを使用して、他のプリンシパルに KMS キーを管理する許可を付与できます。詳細については、「[the section called “デフォルトのキーポリシー”](#)」を参照してください。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子のキーポリシーにキー管理者を追加します "Allow access for Key Administrators"。このステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

15. (オプション) 選択した IAM ユーザーとロールがこの KMS キーを削除しないようにするには、ページの下部にある [Key deletion] (キーの削除) セクションで、[Allow key administrators to delete this key] (キー管理者にこのキーの削除を許可する) のチェックボックスをオフにします。
16. [次へ] を選択します。
17. [暗号化オペレーション](#)で KMS キーを使用できる IAM ユーザーとロールを選択します。

メモ

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子 "Allow use of the key" およびのキーポリシーにキーユーザーを追加します "Allow attachment of persistent resources"。これらのステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

18. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある [Other AWS アカウント] セクションで、[Add another AWS アカウント] を選択し、外部アカウントの AWS アカウント ID 番号を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

Note

外部アカウントでプリンシパルが KMS キーを使用できるようにするには、外部アカウントの管理者が、これらのアクセス許可を付与する IAM ポリシーを作成する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

19. [Next] (次へ) を選択します。
20. キーのキーポリシーステートメントを確認します。キーポリシーを変更するには、[編集](#) を選択します。

21. [次へ] を選択します。
22. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。
23. [Finish] (完了) を選択して HMAC KMS キーを作成します。

AWS KMS API の使用

[CreateKey](#) オペレーションを使用して HMAC KMS キーを作成することができます。以下の例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

HMAC KMS キーを作成するときは、KMS キーのタイプを決定する KeySpec パラメータを指定する必要があります。また、GENERATE_VERIFY_MAC が HMAC キーに唯一有効なキーの用途であっても、GENERATE_VERIFY_MAC の KeyUsage 値を指定する必要があります。[マルチリージョン](#)の HMAC KMS キーを作成するには、値が true の MultiRegion パラメータを追加します。KMS キー作成後にこれらのプロパティを変更することはできません。

CreateKey オペレーションでは、エイリアスを指定することはできませんが、[CreateAlias](#) オペレーションを実行して、新しい KMS キーのエイリアスを作成できます。KMS キーを HMAC キーとして識別するエイリアス (HMAC/test-key など) の使用をお勧めします。これにより、AWS KMS エイリアスでキーをソートおよびフィルタリングできるコンソールで HMAC キーを簡単に識別できますが、キー仕様やキーの使用法では識別できません。

HMAC キーがサポートされていないで AWS リージョン HMAC KMS キーを作成しようとする、CreateKey オペレーションは を返します。UnsupportedOperationException

以下の例では、CreateKey オペレーションを使用して 512 ビットの HMAC KMS キーを作成します。

```
$ aws kms create-key --key-spec HMAC_512 --key-usage GENERATE_VERIFY_MAC
{
  "KeyMetadata": {
    "KeyState": "Enabled",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "Description": "",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": 1669973196.214,
    "MultiRegion": false,
```

```
{
  "KeySpec": "HMAC_512",
  "CustomerMasterKeySpec": "HMAC_512",
  "KeyUsage": "GENERATE_VERIFY_MAC",
  "MacAlgorithms": [
    "HMAC_SHA_512"
  ],
  "AWSAccountId": "111122223333",
  "Origin": "AWS_KMS",
  "Enabled": true
}
```

マルチリージョンのプライマリキーを作成する

[マルチリージョンのプライマリキー](#)は、AWS KMS コンソールまたは AWS KMS API を使用して作成できます。プライマリキーは、がマルチリージョンキー AWS KMS をサポートしている任意の AWS リージョン で作成できます。

マルチリージョンのプライマリキーを作成するには、プリンシパルに、任意の KMS キーを作成するのと[同じアクセス許可](#) (IAM ポリシーの [kms:CreateKey](#) など) が必要です。プリンシパルには [iam:CreateServiceLinkedRole](#) 許可も必要です。[kms:MultiRegionKeyType](#) 条件キーを使用して、マルチリージョンのプライマリキーを作成するアクセス許可を許可または拒否できます。

Note

マルチリージョンのプライマリキーを作成するときは、キーの管理と使用のために選択した IAM ユーザーとロールを慎重に検討してください。IAM ポリシーでは、他の IAM ユーザーおよびロールに、KMS キーを管理するアクセス許可を付与できます。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールの使用

AWS KMS コンソールでマルチリージョンのプライマリキーを作成するには、KMS キーの作成に使用すると同じプロセスを使用します。[Advanced options (アドバンスドオプション)] でマルチリージョンキーを選択します。詳細な手順については、「[KMS キーを作成する](#)」を参照してください。

⚠ Important

エイリアス、説明、またはタグには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。
5. 対称または非対称のキータイプを選択します。デフォルトは [Symmetric] (対称) キーです。

マルチリージョンの対称キーと非対称キーを作成できます。これには、対称であるマルチリージョン HMAC KMS キーが含まれます。

6. キーの用途を選択します。デフォルトは [Encrypt and decrypt] (暗号化および復号化) です。

ヘルプについては、「[KMS キーを作成する](#)」、「[the section called “非対称 KMS キーを作成する”](#)」、または「[the section called “HMAC KMS キーを作成する”](#)」を参照してください。

7. [詳細オプション] を展開します。
8. キーマテリアルオリジンで、プライマリキーとレプリカキーが共有するキーマテリアルを で AWS KMS 生成するには、KMS を選択します。[キーマテリアルをプライマリキーとレプリカキーにインポートする](#)には、[External (Import key material)] (外部 (キーマテリアルのインポート)) を選択します。
9. [リージョンナリティ] 列で、[マルチリージョンキー] を選択します。

KMS キーの作成後は、この設定を変更できません。

10. プライマリキーの[エイリアス](#)を入力します。

エイリアスは、マルチリージョンキーの共有プロパティではありません。マルチリージョンのプライマリキーとそのレプリカに、同じエイリアスまたは異なるエイリアスを指定できます。AWS KMS は、マルチリージョンキーのエイリアスを同期しません。

Note

エイリアスを追加、削除、更新すると、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[エイリアスを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

11. (オプション) プライマリキーの説明を入力します。

説明は、マルチリージョンキーの共有プロパティではありません。マルチリージョンのプライマリキーとそのレプリカに、同じ説明または異なる説明を指定できます。AWS KMS は、マルチリージョンキーのキー説明を同期しません。

12. (オプション) タグキーとオプションのタグ値を入力します。プライマリキーに複数のタグを割り当てるには、[Add tag] (タグを追加する) を選択します。

タグは、マルチリージョンキーの共有プロパティではありません。マルチリージョンのプライマリキーとそのレプリカに同じタグまたは別のタグを割り当てることができます。AWS KMS は、マルチリージョンキーのタグを同期しません。KMS キーのタグはいつでも変更できます。

Note

KMS キーのタグ付けまたはタグ解除により、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[タグを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

13. プライマリキーを管理できる IAM ユーザーとロールを選択します。**メモ**

- このステップでは、[キーポリシー](#)をプライマリキー用に作成するプロセスをスタートします。キーポリシーは、マルチリージョンキーの共有プロパティではありません。マルチリージョンのプライマリキーとそのレプリカに、同じキーポリシーまたは異なるキーポリシーを付与できます。AWS KMS は、マルチリージョンキーのキーポリシーを同期しません。KMS キーのキーポリシーは、いつでも変更できます。
- マルチリージョンのプライマリキーを作成するときは、コンソールで生成された[デフォルトのキーポリシー](#)を使用することを検討してください。このポリシーを変更した場合、コンソールでは、レプリカキーの作成時にキー管理者とユーザーを選択する

ステップは提供されず、対応するポリシーステートメントも追加されません。そのため、これらを手動で追加する必要があります。

- AWS KMS コンソールは、ステートメント識別子のキーポリシーにキー管理者を追加します "Allow access for Key Administrators"。このステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

14. (オプション) 選択した IAM ユーザーとロールがこの KMS キーを削除しないようにするには、ページの下部にある [Key deletion] (キーの削除) セクションで、[Allow key administrators to delete this key] (キー管理者にこのキーの削除を許可する) のチェックボックスをオフにします。
15. [次へ] を選択します。
16. [暗号化オペレーション](#) で KMS キーを使用できる IAM ユーザーとロールを選択します。

メモ

AWS KMS コンソールは、ステートメント識別子 "Allow use of the key" およびのキーポリシーにキーユーザーを追加します "Allow attachment of persistent resources"。これらのステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

17. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある [Other AWS アカウント] セクションで、[Add another AWS アカウント] を選択し、外部アカウントの AWS アカウント ID 番号を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

Note

外部アカウントでプリンシパルが KMS キーを使用できるようにするには、外部アカウントの管理者が、これらのアクセス許可を付与する IAM ポリシーを作成する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

18. [Next] (次へ) を選択します。
19. キーのキーポリシーステートメントを確認します。キーポリシーを変更するには、編集を選択します。
20. [次へ] を選択します。

21. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。
22. 完了 を選択して、マルチリージョンのプライマリキーを作成します。

AWS KMS API の使用

マルチリージョンのプライマリキーを作成するには、[CreateKey](#) オペレーションを使用します。値が True の MultiRegion パラメータを使用します。

たとえば、次のコマンドは、発信者の AWS リージョン (us-east-1) にマルチリージョンのプライマリキーを作成します。キーポリシーを含む、他のすべてのプロパティはデフォルト値を受け入れます。マルチリージョンのプライマリキーのデフォルト値は、他のすべての KMS キーのデフォルト値と同じです ([デフォルトのキーポリシー](#)を含む)。この手順は、対称暗号化キー、デフォルト KMS キーを作成します。

レスポンスには、MultiRegion 要素と典型的なサブ要素を持つ MultiRegionConfiguration 要素、およびレプリカキーを持たないマルチリージョンのプライマリキーの値が含まれます。マルチリージョンキーの [キー ID](#) は、必ず mrk- で始まります。

Important

Description フィールドまたは Tags フィールドには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

```
$ aws kms create-key --multi-region
{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "Enabled": true,
    "KeySpec": "SYMMETRIC_DEFAULT",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "CreationDate": 1606329032.475,
    "Arn": "arn:aws:kms:us-east-1:111122223333:key/mrk-1234abcd12ab34cd56ef1234567890ab",
```

```
    "AWSAccountId": "111122223333",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "MultiRegion": true,
    "MultiRegionConfiguration": {
      "MultiRegionKeyType": "PRIMARY",
      "PrimaryKey": {
        "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "Region": "us-east-1"
      },
      "ReplicaKeys": [ ]
    }
  }
}
```

マルチリージョンのレプリカキーを作成する

マルチリージョンレプリカキーは、AWS KMS コンソール、[ReplicateKey](#) オペレーション、または [AWS::KMS::ReplicaKey](#) AWS CloudFormation テンプレートを使用して作成できます。[CreateKey](#) オペレーションを使用してレプリカキーを作成することはできません。

これらの手順を使用して、[対称暗号化 KMS キー](#)、[非対称 KMS キー](#)、または [HMAC KMS キー](#)などの任意のマルチリージョンプライマリキーをレプリケートすることができます。

このオペレーションが完了すると、新しいレプリカキーは一時的に **Creating** の [キーステータス](#) を持ちます。このキー状態は、新しいレプリカキーの作成プロセスが完了すると、数秒後に **Enabled** に変わります (ただし、[インポートされたキーマテリアル](#) でマルチリージョンキーを作成する場合は **PendingImport** に変わります)。キーステータスが **Creating** の間、キーを管理することはできませんが、暗号化オペレーションで使用することはできません。レプリカキーをプログラムで作成して使用している場合は `KMSInvalidStateException` で再試行するか、[DescribeKey](#) を呼び出して使用前にその `KeyState` 値をチェックします。

レプリカキーを誤って削除した場合は、この手順を使用してレプリカキーを再度作成できます。同じリージョンで同じプライマリキーをレプリケートする場合、作成する新しいレプリカキーは、元のレプリカキーと同じ [共有プロパティ](#) を有します。

⚠ Important

エイリアス、説明、またはタグには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

AWS CloudFormation テンプレートを使用してレプリカキーを作成するには、AWS CloudFormation ユーザーガイドの[AWS::KMS::ReplicaKey](#)」を参照してください。

ステップ 1: レプリカリージョンを選択する

通常、ビジネスモデルと規制要件 AWS リージョン に基づいて、マルチリージョンキーを にレプリケートすることを選択します。例えば、リソースを保管するリージョンにキーをレプリケートできます。または、災害対策の要件に準拠するために、地理的に離れたリージョンにキーをレプリケートすることもできます。

レプリカリージョン AWS KMS の要件は次のとおりです。選択したリージョンがこれらの要件を満たしていない場合、キーレプリケーションの試行は失敗します。

- リージョンごとに 1 つの関連するマルチリージョンキー — プライマリキーと同じリージョンにレプリカキーを作成したり、プライマリキーの別のレプリカと同じリージョンにレプリカキーを作成することはできません。

すでにそのプライマリキーのレプリカがあるリージョンでプライマリキーをレプリケートしようとすると、試行は失敗します。リージョンの現在のレプリカキーが [PendingDeletion キースタタス](#)にある場合は、[レプリカキーの削除をキャンセルする](#)が、レプリカキーが削除されるまで待機します。

- 同じリージョン内の複数の関連しないマルチリージョンキー — 同じリージョン内に、複数の関連しないマルチリージョンキーを持つことができます。例えば、us-east-1 リージョンで 2 つのマルチリージョンのプライマリキーを持つことができます。プライマリキーごとに、us-west-2 リージョンでレプリカキーを持つことができます。
- 同じパーティション内のリージョン — レプリカキーリージョンは、プライマリキーリージョンと同じ [AWS パーティション](#)である必要があります。
- リージョンの有効化 — リージョンが[デフォルトで無効](#)に設定されている場合、AWS アカウントで有効化されるまで、そのリージョンでリソースを作成することはできません。

ステップ 2: レプリカキーを作成する

Note

レプリカキーを作成するときは、レプリカキーの管理と使用のために選択した IAM ユーザーとロールを慎重に検討してください。IAM ポリシーでは、他の IAM ユーザーおよびロールに、KMS キーを管理するアクセス許可を付与できます。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールの使用

AWS KMS コンソールでは、同じオペレーションでマルチリージョンプライマリキーの 1 つ以上のレプリカを作成できます。

この手順は、コンソールでスタンダードの単一リージョン KMS キーを作成する場合と似ています。ただし、レプリカキーはプライマリキーに基づいているため、キー仕様 (対称または非対称)、キー使用法、キーオリジンなどの[共有プロパティ](#)の値は選択しません。

エイリアス、タグ、説明、キーポリシーなど、共有されないプロパティを指定します。便宜上、コンソールにプライマリキーの現在のプロパティ値が表示されますが、変更することもできます。プライマリキー値を保持しても、AWS KMS はこれらの値を同期させません。

Important

エイリアス、説明、またはタグには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。

4. [マルチリージョンのプライマリキー](#)のキー ID またはエイリアスを選択します。KMS キーのキーの詳細ページが開きます。

マルチリージョンのプライマリキーを識別するには、右上隅にあるツールアイコンを使用して [Regionality] (リージョナリティー) 列をテーブルに追加します。

5. [Regionality] (リージョナリティー) タブを選択します。
6. [Related multi-Region keys] (関連するマルチリージョンキー) のセクションで、[Create new replica keys] (新しいレプリカキーの作成) を選択します。

[Related multi-Region keys] (関連するマルチリージョンキー) のセクションには、プライマリキーとそのレプリカキーのリージョンが表示されます。この表示を使用して、新しいレプリカキーのリージョンを選択できます。

7. 1 つ以上の AWS リージョンを選択します。この手順では、選択したリージョンごとにレプリカキーが作成されます。

メニューには、プライマリキーと同じ AWS パーティション内のリージョンのみが含まれます。関連するマルチリージョンキーがすでに存在するリージョンが表示されますが、選択することはできません。メニューのすべてのリージョンに対しては、キーをレプリケートする許可がない場合があります。

リージョンの選択が完了したら、メニューを閉じます。選択したリージョンが表示されます。リージョンへのレプリケーションをキャンセルするには、リージョン名の横にある [X] を選択します。

8. レプリカキーの[エイリアス](#)を入力します。

コンソールには、プライマリキーの現在のエイリアスの 1 つが表示されますが、変更することもできます。マルチリージョンのプライマリキーとそのレプリカに同じエイリアスまたは別のエイリアスを割り当てることができます。エイリアスはマルチリージョンキーの[共有プロパティ](#)ではありません。AWS KMS はマルチリージョンキーのエイリアスを同期しません。

エイリアスを追加、削除、更新すると、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[エイリアスを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

9. (オプション) レプリカキーの説明を入力します。

コンソールには、プライマリキーの現在の説明が表示されますが、変更することもできます。説明は、マルチリージョンキーの共有プロパティではありません。マルチリージョンのプライマ

リキーとそのレプリカに、同じ説明または異なる説明を指定できます。AWS KMS は、マルチリージョンキーのキー説明を同期しません。

10. (オプション) タグキーとオプションのタグ値を入力します。レプリカキーに複数のタグを割り当てるには、[Add tag] (タグを追加する) を選択します。

コンソールには、プライマリキーに現在アタッチされているタグが表示されますが、変更することもできます。タグは、マルチリージョンキーの共有プロパティではありません。マルチリージョンプライマリキーとそのレプリカに同じタグまたは異なるタグを与えることができます。AWS KMS はマルチリージョンキーのタグを同期しません。

KMS キーのタグ付けまたはタグ解除により、KMS キーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[タグを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

11. レプリカキーを管理できる IAM ユーザーとロールを選択します。

メモ

- マルチリージョンプライマリキーの作成時にデフォルトのキーポリシーを変更した場合、コンソールでは、レプリカキーの作成時にキー管理者またはキーユーザーを選択するように求められません (ステップ 11~15)。この場合、キーポリシーの編集ステップ (ステップ 17) で編集を選択して、キー管理者とユーザーに必要なアクセス許可をキーポリシーに手動で追加する必要があります。
- このステップでは、[キーポリシー](#)をレプリカキー用に作成するプロセスをスタートします。コンソールには、プライマリキーの現在のキーポリシーが表示されますが、変更することもできます。キーポリシーは、マルチリージョンキーの共有プロパティではありません。マルチリージョンのプライマリキーとそのレプリカに同じキーポリシーまたは別のキーポリシーを割り当てることができます。AWS KMS は、キーポリシーを同期しません。KMS キーのキーポリシーは、いつでも変更できます。
- AWS KMS コンソールは、ステートメント識別子のキーポリシーにキー管理者を追加します "Allow access for Key Administrators"。このステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

12. (オプション) 選択した IAM ユーザーとロールがこの KMS キーを削除しないようにするには、ページの下部にある [Key deletion] (キーの削除) セクションで、[Allow key administrators to delete this key] (キー管理者にこのキーの削除を許可する) のチェックボックスをオフにします。

13. [次へ] を選択します。
14. [暗号化オペレーション](#)で KMS キーを使用できる IAM ユーザーとロールを選択します。

 メモ

AWS KMS コンソールは、ステートメント識別子 "Allow use of the key" およびのキーポリシーにキーユーザーを追加します "Allow attachment of persistent resources"。これらのステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

15. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある [Other AWS アカウント] セクションで、[Add another AWS アカウント] を選択し、外部アカウントの AWS アカウント ID 番号を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

 Note

外部アカウントでプリンシパルが KMS キーを使用できるようにするには、外部アカウントの管理者が、これらのアクセス許可を付与する IAM ポリシーを作成する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

16. [Next] (次へ) を選択します。
17. キーのキーポリシーステートメントを確認します。キーポリシーを変更するには、編集 を選択します。
18. [次へ] を選択します。
19. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。
20. 完了 を選択して、マルチリージョンレプリカキーを作成します。

AWS KMS API の使用

マルチリージョンのレプリカキーを作成するには、[ReplicateKey](#) オペレーションを使用します。[CreateKey](#) オペレーションを使用してレプリカキーを作成することはできません。このオペレーションでは、一度に 1 つのレプリカキーが作成されます。指定するリージョンは、レプリカキーの[リージョンの要件](#)に準拠している必要があります。

ReplicateKey オペレーションを使用する際、マルチリージョンキーの任意の[共有プロパティ](#)の値は指定しません。共有プロパティ値はプライマリキーからコピーされ、同期が維持されます。ただし、共有されないプロパティには値を指定できます。それ以外の場合、はプライマリキーの値ではなく、KMS キーの標準デフォルト値 AWS KMS を適用します。

Note

Description、KeyPolicy、または Tags パラメータの値を指定しない場合、は空の文字列の説明、[デフォルトのキーポリシー](#)、タグなしのレプリカキー AWS KMS を作成します。Description フィールドまたは Tags フィールドには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

例えば、次のコマンドでは、アジアパシフィック (シドニー) リージョン (ap-southeast-2) にマルチリージョンのレプリカキーを作成します。このレプリカキーは、米国東部 (バージニア北部) リージョン (us-east-1) のプライマリキーをモデルにしています。これは、KeyId パラメータの値で識別されます。この例では、キーポリシーを含む、他のすべてのプロパティのデフォルト値を受け入れます。

レスポンスは新しいレプリカキーを示します。これには、共有プロパティのフィールド (KeyId、KeySpec、KeyUsage、およびキーマテリアルのオリジン (Origin) など) が含まれます。また、プライマリキーとは独立したプロパティも含まれます (Description、キーポリシー (ReplicaKeyPolicy)、タグ (ReplicaTags) など)。

レスポンスには、プライマリキーのキー ARN とリージョン、およびそのすべてのレプリカキー (ap-southeast-2 リージョンで作成されたものを含む) も含まれます。この例では、このプライマリキーが既に欧州 (アイルランド) リージョン (eu-west-1) でレプリケートされていることを、ReplicaKey 要素が示しています。

```
$ aws kms replicate-key \
  --key-id arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \
  --replica-region ap-southeast-2
{
  "ReplicaKeyMetadata": {
    "MultiRegion": true,
    "MultiRegionConfiguration": {
      "MultiRegionKeyType": "REPLICA",
      "PrimaryKey": {
```

```

        "Arn": "arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "Region": "us-east-1"
    },
    "ReplicaKeys": [
        {
            "Arn": "arn:aws:kms:ap-southeast-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
            "Region": "ap-southeast-2"
        },
        {
            "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
            "Region": "eu-west-1"
        }
    ]
},
"AWSAccountId": "111122223333",
"Arn": "arn:aws:kms:ap-southeast-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
"CreationDate": 1607472987.918,
"Description": "",
"Enabled": true,
"KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
"KeyManager": "CUSTOMER",
"KeySpec": "SYMMETRIC_DEFAULT",
"KeyState": "Enabled",
"KeyUsage": "ENCRYPT_DECRYPT",
"Origin": "AWS_KMS",
"CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
"EncryptionAlgorithms": [
    "SYMMETRIC_DEFAULT"
]
},
"ReplicaKeyPolicy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Id\" : \"key-
default-1\",...,
  \"ReplicaTags\": []
}

```

インポートされたキーマテリアルを持つ KMS キーの作成

インポートされたキーマテリアルを使用すると、生成した暗号化キーで AWS リソースを保護できます。次に、AWS KMS にキーマテリアルをインポートする手順の概要を説明します。プロセスの各ステップの詳細については、対応するトピックを参照してください。

1. [キーマテリアルなしで KMS キーを作成](#) – オリジンは EXTERNAL である必要があります。のキーオリジンは、キーがインポートされたキーマテリアル用に設計されており、が KMS キーのキーマテリアルを生成 AWS KMS できない EXTERNAL ことを示します。後のステップで、この KMS キーに独自のキーマテリアルをインポートします。

インポートするキーマテリアルは、関連付けられた AWS KMS キーのキー仕様と互換性がある必要があります。互換性の詳細については、「」を参照してください [the section called “インポートされたキーマテリアルの要件”](#)。

2. [ラッピングパブリックキーとインポートトークンをダウンロード](#) – ステップ 1 を完了した後、ラッピングパブリックキーとインポートトークンをダウンロードします。これらの項目は、インポート中のキーマテリアルを保護します AWS KMS。

このステップでは、RSA ラッピングキーのタイプ (「キー仕様」) と、AWS KMS への転送中のデータの暗号化に使用するラッピングアルゴリズムを選択します。同じキーマテリアルをインポートまたは再インポートするたびに、異なるラッピングキー仕様とラッピングキーアルゴリズムを選択できます。

3. [キーマテリアルを暗号化](#) – ステップ 2 でダウンロードしたラッピングパブリックキーを使用して、独自のシステムで作成したキーマテリアルを暗号化します。
4. [キーマテリアルのインポート](#) – 手順 3 で作成した暗号化されたキーマテリアルと、手順 2 でダウンロードしたインポートトークンをアップロードします。

この段階で、[オプションの有効期限を設定できます](#)。インポートされたキーマテリアルの有効期限が切れると、はそれ AWS KMS を削除し、KMS キーは使用できなくなります。この KMS キーを再度使えるようにするには、同じキーマテリアルを再インポートする必要があります。

インポートオペレーションが正常に完了すると、KMS キーのキーステータスは PendingImport から Enabled に変化します。これで、KMS キーを暗号化オペレーションで使用できます。

AWS KMS は、[KMS キーの作成時](#)、[ラップパブリックキーとインポートトークンのダウンロード時](#)、および [キーマテリアルのインポート時](#)に、AWS CloudTrail ログにエントリを記録します。

は、インポートされたキーマテリアルを削除する時、または が期限切れのキーマテリアルを削除するときに、エントリ AWS KMS も記録します。AWS KMS ???

キーマテリアルをインポートするためのアクセス許可

インポートされたキーマテリアルを使用して KMS キーを作成および管理するには、このプロセスのオペレーションに対するアクセス許可が必要です。KMS キーの作成時、キーポリシーに `kms:GetParametersForImport`、`kms:ImportKeyMaterial`、`kms>DeleteImportedKeyMaterial` のアクセス許可を付与できます。AWS KMS コンソールでは、外部キーマテリアルオリジンを使用してキーを作成すると、これらのアクセス許可がキー管理者に自動的に追加されます。

インポートされたキーマテリアルを持つ KMS キーを作成するには、プリンシパルに次の許可が必要です。

- [kms:CreateKey](#) (IAM ポリシー)
 - インポートされたキーマテリアルを持つ KMS キーに対するこの許可を制限するには、EXTERNAL の値を持つ [kms:KeyOrigin](#) ポリシー条件を使用します。

```
{
  "Sid": "CreateKMSKeysWithoutKeyMaterial",
  "Effect": "Allow",
  "Resource": "*",
  "Action": "kms:CreateKey",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "EXTERNAL"
    }
  }
}
```

- [KMS:GetParametersForImport](#) (キーポリシーまたは IAM ポリシー)
 - 特定のラッピングアルゴリズムとラッピングキー仕様を使用する要求にこのアクセス許可を制限するには、[KMS: WrappingAlgorithm](#) および [KMS: WrappingKeySpec](#) ポリシー条件を使用します。
- [KMS:ImportKeyMaterial](#) (キーポリシーまたは IAM ポリシー)
 - 有効期限が切れるキーマテリアルを許可または禁止し、有効期限を制御するには、[KMS:ExpirationModel](#) および [KMS:Validto](#) ポリシー条件を使用します。

インポートされたキーマテリアルを再インポートするには、プリンシパルに [KMS:getParametersforImport](#) および [KMS:ImportKeyMaterial](#) アクセス許可が必要です。

インポートされたキーマテリアルを削除するには、プリンシパルに [KMS:DeleteImportedKeyMaterial](#) アクセス許可が必要です。

例えば、キーマテリアルがインポートされた KMS キーのすべての側面を管理する KMSAdminRole 許可をサンプルに付与するには、KMS キーのキーポリシーに次のようなキーポリシーステートメントを含めてください。

```
{
  "Sid": "Manage KMS keys with imported key material",
  "Effect": "Allow",
  "Resource": "*",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/KMSAdminRole"
  },
  "Action": [
    "kms:GetParametersForImport",
    "kms:ImportKeyMaterial",
    "kms:DeleteImportedKeyMaterial"
  ]
}
```

インポートされたキーマテリアルの要件

インポートするキーマテリアルには、関連付けられた KMS キーの [キー仕様](#) との互換性が必要です。非対称キーペアの場合、ペアのプライベートキーのみをインポートします。はプライベートキーからパブリックキー AWS KMS を取得します。

AWS KMS は、インポートされたキーマテリアルを持つ KMS キーの次のキー仕様をサポートしています。

- 対称暗号化キー
 - キー仕様：
 - SYMMETRIC_DEFAULT。
 - 要件:
 - 256 ビット (32 バイト) のバイナリデータ。
 - 中国リージョンでは、128 ビット (16 バイト) のバイナリデータである必要があります。

- HMAC キー

- 主な仕様 :

- HMAC_224
 - HMAC_256
 - HMAC_384
 - HMAC_512

- 要件:

- HMAC キーマテリアルは [RFC 2104](#) に準拠している必要があります。
 - キーの長さは、キー仕様で指定された長さと少なくとも同じである必要があります。最大キー長は 1024 ビットです。
 - キーマテリアルが 1024 ビットを超える場合は、キーマテリアルをハッシュし、ハッシュ出力をインポートできます。ハッシュアルゴリズムは、作成する HMAC KMS キーのキー仕様と一致する必要があります。

- 例:

- 2048 ビットのキーマテリアルを HMAC_256 キーにインポートするには、まず 2048 ビットのキーマテリアルの SHA-256 ハッシュを計算し、得られた 256 ビットのハッシュ出力を KMS キーにインポートします。

- 有効なキーの長さ :

- HMAC_224: 224 ~ 1024 ビット
 - HMAC_256: 256 ~ 1024 ビット
 - HMAC_384: 384 ~ 1024 ビット
 - HMAC_512: 512 ~ 1024 ビット

- RSA 非対称プライベートキー

- 主な仕様 :

- RSA_2048
 - RSA_3072
 - RSA_4096

- 要件:

- インポートする RSA 非対称プライベートキーは、[RFC 3447](#) に準拠したキーペアの一部である必要があります。

- 素数の数: 2 (マルチプライム RSA キーはサポートされていません)
- 非対称 キーマテリアルは、[RFC 5208](#) に準拠したパブリックキー暗号標準 (PKCS) #8 形式で BER エンコードまたは DER エンコードされている必要があります。
- 楕円曲線非対称プライベートキー
 - 主な仕様：
 - ECC_NIST_P256 (secp256r1)
 - ECC_NIST_P384 (secp384r1)
 - ECC_NIST_P521 (secp521r1)
 - ECC_SECG_P256K1 (secp256k1)
 - 要件:
 - インポートする ECC 非対称プライベートキーは、[RFC 5915](#) に準拠したキーペアの一部である必要があります。
 - 曲線： NIST P-256、NIST P-384、NIST P-521、または Secp256k1。
 - パラメータ： 名前付き曲線のみ (明示的なパラメータを持つ ECC キーは拒否されます)。
 - パブリックポイント座標： 圧縮、非圧縮、射影のいずれかになります。
 - 非対称 キーマテリアルは、[RFC 5208](#) に準拠したパブリックキー暗号標準 (PKCS) #8 形式で BER エンコードまたは DER エンコードされている必要があります。
- ML-DSA キー
 - 主な仕様：
 - ML_DSA_44
 - ML_DSA_65
 - ML_DSA_87

Important

ML-DSA キーのインポートはサポートされていません。

- SM2 非対称プライベートキー (中国リージョンのみ)
 - 要件:
 - インポートする SM2 非対称プライベートキーは、GM/T 0003 に準拠したキーペアの一部である必要があります。

- パラメータ：名前付き曲線のみ (明示的なパラメータを持つ SM2 キーは拒否されます)。
- パブリックポイント座標：圧縮、非圧縮、射影のいずれかになります。
- 非対称 キーマテリアルは、[RFC 5208](#) に準拠したパブリックキー暗号標準 (PKCS) #8 形式で BER エンコードまたは DER エンコードされている必要があります。

ステップ 1: キーマテリアルなしで AWS KMS key を作成する

デフォルトでは、は KMS キーを作成するときにキーマテリアル AWS KMS を作成します。代わりに独自のキーマテリアルをインポートするには、最初にキーマテリアルなしで KMS キーを作成します。そして、キーマテリアルをインポートします。キーマテリアルなしで KMS キーを作成するには、AWS KMS コンソールまたは [CreateKey](#) オペレーションを使用します。

キーマテリアルなしでキーを作成するには、[オリジン](#)として EXTERNAL を指定します。KMS キーのオリジンプロパティは変更不可です。一度作成すると、インポートされたキーマテリアル用に設計された KMS キーを、AWS KMS または他のソースからのキーマテリアルを持つ KMS キーに変換することはできません。

EXTERNAL オリジンを持つ KMS キーの[キーステータス](#)であり、PendingImport であるキーマテリアルはありません。KMS キーは、無期限に PendingImport 状態を維持できます。ただし、暗号化オペレーションでは PendingImport 状態で KMS キーを使用することはできません。キーマテリアルをインポートすると、KMS キーのキーステータスが Enabled に変わり、暗号オペレーションで使えるようになります。

AWS KMS は、[KMS キーの作成](#)、[パブリックキーとインポートトークンのダウンロード](#)、[キーマテリアルのインポート](#)時に AWS CloudTrail ログにイベントを記録します。AWS KMS は、[インポートされたキーマテリアルの削除](#)時、または [が期限切れのキーマテリアルを削除](#)するときに CloudTrail イベントも記録します。AWS KMS [???](#)

トピック

- [キーマテリアルなしで KMS キーを作成する \(コンソール\)](#)
- [キーマテリアルなしで KMS キーを作成する \(AWS KMS API\)](#)

キーマテリアルなしで KMS キーを作成する (コンソール)

インポートされたキーマテリアルの KMS キーを 1 回作成する必要があるだけです。同じキーマテリアルを既存の KMS キーに必要な回数だけインポートおよび再インポートできますが、1 つの KMS

キーに別のキーマテリアルをインポートすることはできません。詳細については、「[ステップ 2: ラップパブリックキーおよびインポートトークンのダウンロード](#)」を参照してください。

キーマテリアルがインポートされた既存の KMS キーを、[Customer managed keys] (カスタマー管理型キー) テーブルで検索するには、右上隅にある歯車アイコンを使用して、KMS キーのリストの [Origin] (オリジン) 列を表示します。インポートされたキーの Origin 値は EXTERNAL (キーマテリアルのインポート) です。

インポートされたキーマテリアルを持つ KMS キーを作成するには、まず[目的のキータイプの KMS キー作成手順](#)に従ってください。ただし、以下の例外事項があります。

キーの使用方法を選択したら、次の操作を行います。

1. [詳細オプション] を展開します。
2. [キーマテリアルのオリジン] で、[EXTERNAL (キーマテリアルのインポート)] を選択します。
3. インポートされたキーマテリアルの使用による影響について理解したことを示すため、[インポートされたキーの使用によるセキュリティと耐久性への影響について理解しました] の隣にあるチェックボックスをオンにします。これらの意味については、「[インポートされたキーマテリアルの保護](#)」を参照してください。
4. オプション: インポートされたキーマテリアルを持つ [マルチリージョン KMS キー](#) を作成するには、[リージョンナリティ] で [マルチリージョンキー] を選択します。
5. 基本的な手順に戻ります。基本的な手順の残りのステップは、そのタイプの KMS キーについてすべて同じです。

[完了] を選択すると、キーマテリアルがなく、ステータス ([キーステータス](#)) が [インポート保留中] の KMS キーが作成されたことになります。

ただし、[カスタマー管理型のキー] テーブルに戻るのではなく、コンソールには、キーマテリアルのインポートに必要なパブリックキーとインポートトークンをダウンロードできるページが表示されます。ここでダウンロードのステップを続行することも、[キャンセル] を選択してこの時点で停止することもできます。いつでもこのダウンロードのステップに戻ることができます。

次の手順: [ステップ 2: ラップパブリックキーおよびインポートトークンのダウンロード](#)

キーマテリアルなしで KMS キーを作成する (AWS KMS API)

[AWS KMS API](#) を使用してキーマテリアルを持たない対称暗号化 KMS キーを作成するには、Origin パラメータが EXTERNAL に設定された [CreateKey](#) リクエストを送信します。次の例で

は、[AWS Command Line Interface \(AWS CLI\)](#) を使用してこのオペレーションを行う方法を示します。

```
$ aws kms create-key --origin EXTERNAL
```

コマンドが成功した場合は、以下のような出力が表示されます。AWS KMS キーの Origin は EXTERNAL で、 は KeyState です PendingImport。

 Tip

コマンドが成功しない場合は、KMSInvalidStateException または NotFoundException が表示されることがあります。リクエストは再試行できます。

```
{
  "KeyMetadata": {
    "Origin": "EXTERNAL",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Description": "",
    "Enabled": false,
    "MultiRegion": false,
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "PendingImport",
    "CreationDate": 1568289600.0,
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

後のステップで使用するために、コマンド出力からの KeyId 値をコピーしてから、「[ステップ 2: ラップパブリックキーおよびインポートトークンのダウンロード](#)」に進みます。

Note

このコマンドは、SYMMETRIC_DEFAULT の KeySpec および ENCRYPT_DECRYPT の KeyUsage を含む対称暗号化 KMS キーを作成します。オプションのパラメータ `--key-spec` と `--key-usage` を使用して、非対称 KMS キーまたは HMAC KMS キーを作成できます。詳細については、[CreateKey](#) オペレーションを参照してください。

ステップ 2: ラップパブリックキーおよびインポートトークンのダウンロード

[キーマテリアル AWS KMS key なしでを作成](#)したら、AWS KMS コンソールまたは [GetParametersForImport](#) API を使用して、その KMS キーのラップパブリックキーとインポートトークンをダウンロードします。ラップパブリックキーとインポートトークンはセットであり分割できないので、組み合わせて使用する必要があります。

ラップパブリックキーは、転送用の[キーマテリアルを暗号化](#)するために使用します。RSA ラッピングキーペアをダウンロードする前に、RSA ラッピングキーペアの長さ (キー仕様) と、[ステップ 3](#) でインポートしたキーマテリアルを転送用に暗号化するために使用するラッピングアルゴリズムを選択します。は SM2 ラッピングキー仕様 AWS KMS もサポートしています (中国リージョンのみ)。

各ラップパブリックキーとインポートトークンのセットは 24 時間有効です。それらを 24 時間以内にキーマテリアルのインポートに使用しなかった場合、新しいセットをダウンロードする必要があります。ラップパブリックキーとインポートトークンの新しいセットは、任意のタイミングでダウンロードが可能です。これにより、RSA ラップキーの長さ (キー仕様) を変更したり、紛失したセットを置き換えたりできます。

また、ラップパブリックキーとインポートトークンのセットをダウンロードして、KMS キーに[同じキーマテリアルを再インポート](#)することもできます。このオペレーションは、キーマテリアルの有効期限を設定もしくは変更したり、期限切れまたは削除済みのキーマテリアルを復元したりする場合に実行します。キーマテリアルは、インポートするたびにダウンロードして再暗号化する必要があります AWS KMS。

ラップパブリックキーの使用

ダウンロードには、ラップパブリックキーとも呼ばれる AWS アカウント、に固有のパブリックキーが含まれています。

キーマテリアルをインポートする前に、パブリックラッピングキーを使用してキーマテリアルを暗号化し、暗号化されたキーマテリアルを にアップロードします AWS KMS。は暗号化されたキーマテリアル AWS KMS を受信すると、対応するプライベートキーを使用してキーマテリアルを復号し、AES 対称キーでキーマテリアルを再暗号化します。すべて AWS KMS ハードウェアセキュリティモジュール (HSM) 内で行われます。

インポートトークンの使用

ダウンロードには、キーマテリアルが正しくインポートされたことを保証するメタデータが付随する、インポートトークンが含まれています。暗号化されたキーマテリアルを にアップロードするときは AWS KMS、このステップでダウンロードしたのと同じインポートトークンをアップロードする必要があります。

ラップパブリックキーの仕様を選択

インポート中にキーマテリアルを保護するには、ダウンロード元のラップパブリックキー AWS KMS とサポートされている [ラップアルゴリズム](#) を使用して暗号化します。ラップパブリックキーとインポートトークンをダウンロードする前に、キー仕様を選択します。すべてのラッピングキーペアは、AWS KMS ハードウェアセキュリティモジュール (HSMs)。プレーンテキストのプライベートキーが、HSM の外部に出ることはありません。

RSA ラップキーの仕様

ラップパブリックキーのキー仕様によって、AWS KMS への転送時にキーマテリアルを保護する RSA キーペアにおける、キーの長さが決まります。一般的には、実用的で最長のラップパブリックキーを使用することをお勧めします。各種の HSM やキーマネージャーをサポートするために、いくつかのラップパブリックキー仕様が提供されています。

AWS KMS は、注記されている場合を除き、すべてのタイプのキーマテリアルをインポートするために使用される RSA ラッピングキーについて、次のキー仕様をサポートしています。

- RSA_4096 (推奨)
- RSA_3072
- RSA_2048

Note

ECC_NIST_P521 キーマテリアル、RSA_2048 パブリックラップキー仕様、および RSAES_OAEP_SHA_* ラップアルゴリズムの組み合わせはサポートされていません。

RSA_2048 パブリックラップキーを使用して、ECC_NIST_P521 のキーマテリアルを直接ラップすることはできません。大きなラップキー、または RSA_AES_KEY_WRAP_SHA_* ラップアルゴリズムを使用してください。

SM2 ラップキーの仕様 (中国リージョンのみ)

AWS KMS は、非対称キーマテリアルのインポートに使用される SM2 ラッピングキーについて、次のキー仕様をサポートしています。

- SM2

ラップアルゴリズムの選択

インポート中にキーマテリアルを保護するには、ダウンロードしたラップパブリックキーと、サポートされているラップアルゴリズムを使用して、キーマテリアルを暗号化します。

AWS KMS は、いくつかの標準 RSA ラッピングアルゴリズムと 2 ステップハイブリッドラッピングアルゴリズムをサポートしています。基本的には、インポートしたキーマテリアルおよび[ラップキー仕様](#)との互換性がある、最も安全なラップアルゴリズムを使用することをお勧めします。通常、ハードウェアセキュリティモジュール (HSM) がサポートするアルゴリズム、またはキーマテリアルを保護するキー管理システムを選択します。

次の表は、キーマテリアルと KMS キーの各タイプでサポートされているラップアルゴリズムを示しています。これらのアルゴリズムは、優先度順にリストされています。

キーマテリアル	サポートされるラップアルゴリズムと仕様
対称暗号化キー	ラップアルゴリズム:
256 ビット AES キー	RSAES_OAEP_SHA_256
128 ビット SM4 キー (中国リージョンのみ)	RSAES_OAEP_SHA_1
	非推奨となっているラップアルゴリズム:
	RSAES_PKCS1_V1

 **Note**
2023 年 10 月 10 日 AWS KMS 現在、は RSAES_PKCS1_V1_5 ラッ

キーマテリアル	サポートされるラップアルゴリズムと仕様
	プアルゴリズムをサポートしていません。 ラップキーの仕様: RSA_2048 RSA_3072 RSA_4096
非対称 RSA プライベートキー	ラップアルゴリズム: RSA_AES_KEY_WRAP_SHA_256 RSA_AES_KEY_WRAP_SHA_1 SM2PKE (中国リージョンのみ) ラップキーの仕様: RSA_2048 RSA_3072 RSA_4096 SM2 (中国リージョンのみ)

キーマテリアル	サポートされるラップアルゴリズムと仕様
非対称楕円曲線 (ECC) プライベートキー RSA_2048 ラップキー仕様の RSAES_OAEP_SHA_* ラップアルゴリズムを使用して、ECC_NIST_P521 のキーマテリアルをラップすることはできません。	ラップアルゴリズム: RSA_AES_KEY_WRAP_SHA_256 RSA_AES_KEY_WRAP_SHA_1 RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 SM2PKE (中国リージョンのみ) ラップキーの仕様: RSA_2048 RSA_3072 RSA_4096 SM2 (中国リージョンのみ)
非対称 SM2 プライベートキー (中国リージョンのみ)	ラップアルゴリズム: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 SM2PKE (中国リージョンのみ) ラップキーの仕様: RSA_2048 RSA_3072 RSA_4096 SM2 (中国リージョンのみ)

キーマテリアル	サポートされるラップアルゴリズムと仕様
HMAC キー	ラップアルゴリズム: RSAES_OAEP_SHA_256 RSAES_OAEP_SHA_1 ラップキーの仕様: RSA_2048 RSA_3072 RSA_4096

 Note

RSA_AES_KEY_WRAP_SHA_256 と RSA_AES_KEY_WRAP_SHA_1 のラッピングアルゴリズムは、中国リージョンではサポートされていません。

- RSA_AES_KEY_WRAP_SHA_256 – 生成した AES 対称キーでキーマテリアルを暗号化し、その後、ダウンロードした RSA パブリックラップキーと RSAES_OAEP_SHA_256 ラップアルゴリズムにより AES 対称キーを暗号化する、2 段階のハイブリッドラップアルゴリズム。

RSA プライベートキーマテリアルをラッピングするには、RSA_AES_KEY_WRAP_SHA_* ラッピングアルゴリズムが必要です。ただし、SM2PKE ラッピングアルゴリズムを使用する必要がある中国リージョンは除きます。

- RSA_AES_KEY_WRAP_SHA_1 – 生成した AES 対称キーでキーマテリアルを暗号化し、その後、ダウンロードした RSA ラップパブリックキーと RSAES_OAEP_SHA_1 ラップアルゴリズムにより AES 対称キーを暗号化する、2 段階のハイブリッドラップアルゴリズム。

RSA プライベートキーマテリアルをラッピングするには、RSA_AES_KEY_WRAP_SHA_* ラッピングアルゴリズムが必要です。ただし、SM2PKE ラッピングアルゴリズムを使用する必要がある中国リージョンは除きます。

- RSAES_OAEP_SHA_256 — SHA -256 ハッシュ関数を使用した最適な非対称暗号化パディング (OAEP) を使用する RSA 暗号化アルゴリズム。

- RSAES_OAEP_SHA_1 — SHA-1 ハッシュ関数を使用した最適な非対称暗号化パディング (OAEP) を使用する RSA 暗号化アルゴリズム。
- RSAES_PKCS1_V1_5 (廃止。2023 年 10 月 10 日現在、AWS KMS は RSAES_PKCS1_V1_5 ラップアルゴリズムをサポートしていません) – PKCS #1 バージョン 1.5 で定義されたパディング形式を持つ RSA 暗号化アルゴリズム。
- SM2PKE (中国リージョンのみ) – GM/T 0003.4-2012 の OSCCA によって定義される楕円曲線ベースの暗号化アルゴリズムです。

トピック

- [ラップパブリックキーとインポートトークンのダウンロード \(コンソール\)](#)
- [ラップパブリックキーとインポートトークンのダウンロード \(AWS KMS API\)](#)

ラップパブリックキーとインポートトークンのダウンロード (コンソール)

AWS KMS コンソールを使用して、ラップパブリックキーとインポートトークンをダウンロードできます。

1. [キーマテリアルなしで KMS キーを作成するステップ](#)を完了し、ラップキーとインポートトークンのダウンロードのページが開いている場合は、[Step 9](#)に進みます。
2. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
3. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
4. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。

Tip

キーマテリアルは、EXTERNAL (キーマテリアルのインポート) の Origin を持つ KMS キーにのみインポートできます。これは、キーマテリアルなしで KMS キーが作成されたことを示します。テーブルに [オリジン] 列を追加するには、右上隅の設定アイコン



を選択します。[オリジン] をオンにして、[確認] を選択します。

5. インポート保留中の KMS キーのエイリアスまたはキー ID を選択します。
6. [暗号構成] タブを選択し、その値を表示します。これらのタブは、[General configuration] (一般設定) セクションの下にあります。

キーマテリアルは、EXTERNAL (キーマテリアルのインポート) の Origin として、KMS キーにのみインポートできます。インポートされたキーマテリアルで KMS キーを作成する方法の詳細については、[キーの AWS KMS キーマテリアルのインポート](#) を参照してください。

7. [キーマテリアル] タブを選択し、[キーマテリアルをインポート] を選択します。

[キーマテリアル] タブは、EXTERNAL (キーマテリアルのインポート) の Origin 値が指定された KMS キー に対してのみ表示されます。

8. [ラップキー仕様の選択] で、使用する KMS キーの設定を選択します。このキーの作成後は、キー仕様を変更することはできません。
9. [ラップアルゴリズムの選択] で、キーマテリアルの暗号化に使用するオプションを選択します。オプションの詳細については、「[ラップアルゴリズムの選択](#)」を参照してください。
10. [ラップパブリックキーとインポートトークンをダウンロード] を選択した後、ファイルを保存します。

[次へ] オプションがある場合、今すぐプロセスを続行するには、[次へ] を選択します。後で続行するには、[キャンセル] を選択します。

11. 前のステップ (Import_Parameters_<key_id>_<timestamp>) で保存した .zip ファイルを解凍します。

フォルダには以下のファイルが含まれています。

- WrappingPublicKey.bin という名前のファイルにある、ラップパブリックキー。
- ImportToken.bin という名前のファイルにある、インポートトークン。
- README.txt という名前のテキストファイル。このファイルには、ラップパブリックキー、キーマテリアルの暗号化に使用するラップアルゴリズム、およびラップパブリックキーとインポートトークンの有効期限が切れる日時に関する情報が格納されています。

12. プロセスを続行する場合は、「[キーマテリアルの暗号化](#)」を参照してください。

ラップパブリックキーとインポートトークンのダウンロード (AWS KMS API)

パブリックキーとインポートトークンのダウンロードには、[GetParametersForImport](#) API を使用します。インポートされたキーマテリアルに関連付けられる KMS キーを指定します。この KMS キーには、EXTERNAL の [Origin](#) 値が必要です。

Note

ML-DSA KMS キーのキーマテリアルをインポートすることはできません。

この例では、RSA_AES_KEY_WRAP_SHA_256 ラップアルゴリズム、RSA_3072 ラップパブリックキーの仕様、およびサンプルキー ID を規定します。これらのサンプル値は、実際のダウンロードのための有効な値に置き換えます。このオペレーションでは、キーの識別子として [キー ID](#) または [キー ARN](#) を使用できますが、[エイリアス名](#)や[エイリアス ARN](#) を使用することはできません。

```
$ aws kms get-parameters-for-import \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --wrapping-algorithm RSA_AES_KEY_WRAP_SHA_256 \  
  --wrapping-key-spec RSA_3072
```

コマンドが成功した場合は、以下のような出力が表示されます。

```
{  
  "ParametersValidTo": 1568290320.0,  
  "PublicKey": "public key (base64 encoded)",  
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
  "ImportToken": "import token (base64 encoded)"  
}
```

次のステップのためにデータを準備します。パブリックキーとインポートトークンを base64 でデコードし、デコードされた値をファイルに保存します。

パブリックキーとインポートトークンを base64 でデコードするには、次を実行します。

1. base64 でエンコードされたパブリックキー (この例の出力では ##### (base64 #####)) をコピーして、新しいファイルに貼り付けて、ファイルを保存します。ファイルに PublicKey.b64 などの名前を付けます。
2. ファイル内容の base64 デコード、およびデコードされたデータの新しいファイルへの保存に、[OpenSSL](#) を使用します。次の例では、前のステップ (PublicKey.b64) で保存したファイルのデータをデコードして、WrappingPublicKey.bin という名前の新しいファイルに出力を保存します。

```
$ openssl enc -d -base64 -A -in PublicKey.b64 -out WrappingPublicKey.bin
```

- base64 でエンコードされたインポートトークン (この例の出力では #####
(base64 #####)) をコピーして、新しいファイルに貼り付けて、ファイルを保存します。importtoken.b64 など、ファイルにわかりやすい名前を付けます。
- ファイル内容の base64 デコード、およびデコードされたデータの新しいファイルへの保存に、[OpenSSL](#) を使用します。次の例では、前のステップ (ImportToken.b64) で保存したファイルのデータをデコードして、ImportToken.bin という名前の新しいファイルに出力を保存します。

```
$ openssl enc -d -base64 -A -in importtoken.b64 -out ImportToken.bin
```

[ステップ 3: キーマテリアルを暗号化する](#) に進みます。

ステップ 3: キーマテリアルを暗号化する

[パブリックキーとインポートトークンをダウンロード](#) した後、ダウンロードしたパブリックキーと指定したラップアルゴリズムを使用してキーマテリアルを暗号化します。パブリックキーまたはインポートトークンを置き換える必要がある場合、あるいはラップアルゴリズムを変更する必要がある場合は、新しいパブリックキーとインポートトークンをダウンロードする必要があります。が AWS KMS サポートするパブリックキーとラップアルゴリズムの詳細については、[ラップパブリックキーの仕様を選択](#)「」および「」を参照してください[ラップアルゴリズムの選択](#)。

キーマテリアルはバイナリ形式である必要があります。詳細については、「[インポートされたキーマテリアルの要件](#)」を参照してください。

Note

非対称キーペアの場合、プライベートキーのみを暗号化してインポートします。は、プライベートキーからパブリックキー AWS KMS を取得します。

ECC_NIST_P521 キーマテリアル、RSA_2048 パブリックラップキー仕様、および RSAES_OAEP_SHA_* ラップアルゴリズムの組み合わせはサポートされていません。

RSA_2048 パブリックラップキーを使用して、ECC_NIST_P521 のキーマテリアルを直接ラップすることはできません。大きなラップキー、または RSA_AES_KEY_WRAP_SHA_* ラップアルゴリズムを使用してください。

RSA_AES_KEY_WRAP_SHA_256 および RSA_AES_KEY_WRAP_SHA_1 ラッピングアルゴリズムは、中国リージョンではサポートされていません。

通常、ハードウェアセキュリティモジュール (HSM) またはキー管理システムからエクスポートする場合、キーマテリアルを暗号化します。バイナリ形式でキーマテリアルをエクスポートする方法については、HSM またはキー管理システムに関するドキュメントを参照してください。OpenSSL を使用して、概念実証デモを提供する、次のセクションを参照できます。

キーマテリアルを暗号化する場合、[パブリックキーとインポートトークンをダウンロード](#)したときに指定した、同じラップアルゴリズムを使用します。指定したラップアルゴリズムを見つけるには、関連付けられた [GetParametersForImport](#) リクエストの CloudTrail ログイベントを確認してください。

テスト用のキーマテリアルを生成

次の OpenSSL コマンドは、サポートされている各タイプのテスト用のキーマテリアルを生成します。これらの例は、テストと概念実証デモのみを目的としています。本稼働システムの場合、ハードウェアセキュリティモジュールやキー管理システムなど、より安全な方法を使用してキーマテリアルを生成します。

非対称キーペアのプライベートキーを DER でエンコードされた形式に変換するには、パイプを使ってキーマテリアル生成コマンドを次の `openssl pkcs8` コマンドに渡します。この `topk8` パラメータは、プライベートキーを入力として受け取り、PKCS#8 形式のキーを返すように OpenSSL に指示します。(デフォルトの動作は逆です)

```
openssl pkcs8 -topk8 -outform der -nocrypt
```

次のコマンドは、サポートされている各キータイプのテストキーマテリアルを生成します。

- 対称暗号化キー (32 バイト)

このコマンドは、256 ビット対称キー (32 バイトのランダム文字列) を生成し、`PlaintextKeyMaterial.bin` ファイルに保存します。このキーマテリアルをエンコードする必要はありません。

```
openssl rand -out PlaintextKeyMaterial.bin 32
```

中国リージョンでのみ、128 ビット対称キー (16 バイトのランダム文字列) を生成する必要があります。

```
openssl rand -out PlaintextKeyMaterial.bin 16
```

- HMAC キー

このコマンドは、指定したサイズのランダムバイト文字列を生成します。このキーマテリアルをエンコードする必要はありません。

HMAC キーの長さは、KMS キーのキー仕様で定義されている長さと一致する必要があります。例えば、KMS キーが HMAC_384 の場合、384 ビット (48 バイト) キーをインポートする必要があります。

```
openssl rand -out HMAC_224_PlaintextKey.bin 28
```

```
openssl rand -out HMAC_256_PlaintextKey.bin 32
```

```
openssl rand -out HMAC_384_PlaintextKey.bin 48
```

```
openssl rand -out HMAC_512_PlaintextKey.bin 64
```

- RSA プライベートキー

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:2048 | openssl pkcs8 -topk8 -outform der -nocrypt > RSA_2048_PrivateKey.der
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:3072 | openssl pkcs8 -topk8 -outform der -nocrypt > RSA_3072_PrivateKey.der
```

```
openssl genpkey -algorithm rsa -pkeyopt rsa_keygen_bits:4096 | openssl pkcs8 -topk8 -outform der -nocrypt > RSA_4096_PrivateKey.der
```

- ECC プライベートキー

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-256 | openssl pkcs8 -topk8 -outform der -nocrypt > ECC_NIST_P256_PrivateKey.der
```

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-384 | openssl pkcs8 -topk8 -outform der -nocrypt > ECC_NIST_P384_PrivateKey.der
```

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:P-521 | openssl pkcs8 -topk8 -outform der -nocrypt > ECC_NIST_P521_PrivateKey.der
```

```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:secp256k1 | openssl pkcs8 -topk8 -outform der -nocrypt > ECC_SECG_P256K1_PrivateKey.der
```

- SM2 プライベートキー (中国リージョンのみ)


```
openssl genpkey -algorithm ec -pkeyopt ec_paramgen_curve:sm2 | openssl pkcs8 -topk8 -outform der -nocrypt > SM2_PrivateKey.der
```

OpenSSL によるキーマテリアルの暗号化の例

以下の例は、[OpenSSL](#) を使用して、ダウンロードしたパブリックキーでキーマテリアルを暗号化する方法を示しています。SM2 パブリックキー (中国リージョンのみ) を使用してキーマテリアルを暗号化するには、[SM2OfflineOperationHelper クラス](#) を使用します。各ラッピングアルゴリズムがサポートするキーマテリアルタイプの詳細については、「[the section called “ラップアルゴリズムの選択”](#)」を参照してください。

Important

これらの例では、概念実証デモのみです。本稼働システムの場合、より安全な方法 (商用 HSM またはキー管理システムなど) を使用して、キーマテリアルを生成し、保存します。ECC_NIST_P521 キーマテリアル、RSA_2048 パブリックラップキー仕様、および RSAES_OAEP_SHA_* ラップアルゴリズムの組み合わせはサポートされていません。RSA_2048 パブリックラップキーを使用して、ECC_NIST_P521 のキーマテリアルを直接ラップすることはできません。大きなラップキー、または RSA_AES_KEY_WRAP_SHA_* ラップアルゴリズムを使用してください。

RSAES_OAEP_SHA_1

AWS KMS は、対称暗号化キー (SYMMETRIC_DEFAULT)、楕円曲線 (ECC) プライベートキー、SM2 プライベートキー、および HMAC キーの RSAES_OAEP_SHA_1 をサポートしています。

RSAES_OAEP_SHA_1 は RSA プライベートキーではサポートされていません。また、任意の RSAES_OAEP_SHA_* ラップアルゴリズムの RSA_2048 パブリックラップキーを使用して ECC_NIST_P521 (secp521r1) プライベートキーをラップすることはできません。より大きいサイズのパブリックラップキーまたは RSA_AES_KEY_WRAP ラップアルゴリズムを使用する必要があります。

次の例では、[ダウンロードしたパブリックキー](#)と RSAES_OAEP_SHA_1 ラップアルゴリズムを使用してキーマテリアルを暗号化し、EncryptedKeyMaterial.bin ファイルに保存します。

この例では、以下のようにになっています：

- *WrappingPublicKey.bin* は、ダウンロードしたラップパブリックキーを含むファイルです。
- *PlaintextKeyMaterial.bin* は、PlaintextKeyMaterial.bin、HMAC_384_PlaintextKey.bin、暗号化するキーマテリアルを含むファイルです。

```
$ openssl pkeyutl \  
-encrypt \  
-in PlaintextKeyMaterial.bin \  
-out EncryptedKeyMaterial.bin \  
-inkey WrappingPublicKey.bin \  
-keyform DER \  
-pubin \  
-pkeyopt rsa_padding_mode:oaep \  
-pkeyopt rsa_oaep_md:sha1
```

RSAES_OAEP_SHA_256

AWS KMS は、対称暗号化キー (SYMMETRIC_DEFAULT)、楕円曲線 (ECC) プライベートキー、SM2 プライベートキー、および HMAC キーの RSAES_OAEP_SHA_256 をサポートしています。

RSAES_OAEP_SHA_256 は RSA プライベートキーではサポートされていません。また、任意の RSAES_OAEP_SHA_* ラップアルゴリズムの RSA_2048 パブリックラップキーを使用して ECC_NIST_P521 (secp521r1) プライベートキーをラップすることはできません。より大きいサイズのパブリックキーまたは RSA_AES_KEY_WRAP ラップアルゴリズムを使用する必要があります。

次の例では、[ダウンロードしたパブリックキー](#)と RSAES_OAEP_SHA_256 ラップアルゴリズムを使用してキーマテリアルを暗号化し、EncryptedKeyMaterial.bin ファイルに保存します。

この例では、以下のようにになっています：

- *WrappingPublicKey.bin* は、ダウンロードしたパブリックラップキーを含むファイルです。コンソールからパブリックキーをダウンロードした場合、このファイルの名前は wrappingKey_KMS key_key_ID_timestamp (例えば、wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909) となります。
- *PlaintextKeyMaterial.bin* は、PlaintextKeyMaterial.bin、HMAC_384_PlaintextKey.bin、暗号化するキーマテリアルを含むファイルです。

```
$ openssl pkeyutl \  
-encrypt \  
-in PlaintextKeyMaterial.bin \  
-out EncryptedKeyMaterial.bin \  
-inkey WrappingPublicKey.bin \  
-keyform DER \  
-pubin \  
-pkeyopt rsa_padding_mode:oaep \  
-pkeyopt rsa_oaep_md:sha256 \  
-pkeyopt rsa_mgf1_md:sha256
```

RSA_AES_KEY_WRAP_SHA_1

RSA_AES_KEY_WRAP_SHA_1 ラップアルゴリズムには 2 つの暗号化オペレーションオペレーションが含まれています。

1. 生成した AES 対称キーと AES 対称暗号化アルゴリズムを使用してキーマテリアルを暗号化します。
2. 使用した AES 対称キーを、ダウンロードしたパブリックキーと RSAES_OAEP_SHA_1 ラップアルゴリズムで暗号化します。

RSA_AES_KEY_WRAP_SHA_1 ラップアルゴリズムには OpenSSL バージョン 3.x 以降が必要です。

1. 256 ビット AES 対称暗号化キーを生成する

このコマンドは、256 ランダムビットで構成される AES 対称暗号化キーを生成し、aes-key.bin ファイルに保存します

```
# Generate a 32-byte AES symmetric encryption key  
$ openssl rand -out aes-key.bin 32
```

2. AES 対称暗号化キーを使用してキーマテリアルを暗号化する

このコマンドは、AES 対称暗号化キーを使用してキーマテリアルを暗号化し、暗号化されたキーマテリアルを key-material-wrapped.bin ファイルに保存します。

このコマンド例では、

- *PlaintextKeyMaterial.bin* は、PlaintextKeyMaterial.bin、HMAC_384_PlaintextKeyMaterial.bin、インポートするキーマテリアルを含むファイルです。
- *aes-key.bin* は、前のコマンドで生成した 256 ビット AES 対称暗号化キーを含むファイルです。

```
# Encrypt your key material with the AES symmetric encryption key
$ openssl enc -id-aes256-wrap-pad \
  -K "$(xxd -p < aes-key.bin | tr -d '\n')" \
  -iv A65959A6 \
  -in PlaintextKeyMaterial.bin \
  -out key-material-wrapped.bin
```

3. AES 対称暗号化キーをパブリックキーで暗号化する

このコマンドは、ダウンロードしたパブリックキーと RSAES_OAEP_SHA_1 ラップアルゴリズムで AES 対称暗号化キーを暗号化し、DER エンコードして aes-key-wrapped.bin ファイルに保存します。

このコマンド例では、

- *WrappingPublicKey.bin* は、ダウンロードしたパブリックラップキーを含むファイルです。コンソールからパブリックキーをダウンロードした場合、このファイルの名前は wrappingKey_KMS_key_key_ID_timestamp (例えば、wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909) となります
- *aes-key.bin* は、このシーケンスの例の最初のコマンドで生成した 256 ビット AES 対称暗号化キーを含むファイルです。

```
# Encrypt your AES symmetric encryption key with the downloaded public key
$ openssl pkeyutl \
  -encrypt \
  -in aes-key.bin \
  -out aes-key-wrapped.bin \
  -inkey WrappingPublicKey.bin \
  -keyform DER \
  -pubin \
  -pkeyopt rsa_padding_mode:oaep \
  -pkeyopt rsa_oaep_md:sha1 \
```

```
-pkeyopt rsa_mgf1_md:sha1
```

4. インポートするファイルを生成する

暗号化されたキーマテリアルを含むファイルと暗号化された AES キーを含むファイルを連結します。これらを EncryptedKeyMaterial.bin ファイルに保存します。このファイルは、[ステップ 4: キーマテリアルのインポート](#) にインポートするファイルです。

このコマンド例では、

- *key-material-wrapped.bin* は、暗号化されたキーマテリアルを含むファイルです。
- *aes-key-wrapped.bin* は、暗号化された AES 暗号化キーを含むファイルです。

```
# Combine the encrypted AES key and encrypted key material in a file
$ cat aes-key-wrapped.bin key-material-wrapped.bin > EncryptedKeyMaterial.bin
```

RSA_AES_KEY_WRAP_SHA_256

RSA_AES_KEY_WRAP_SHA_256 ラップアルゴリズムには 2 つの暗号化手順が含まれています。

1. 生成した AES 対称キーと AES 対称暗号化アルゴリズムを使用してキーマテリアルを暗号化します。
2. 使用した AES 対称キーを、ダウンロードしたパブリックキーと RSAES_OAEP_SHA_256 ラップアルゴリズムで暗号化します。

RSA_AES_KEY_WRAP_SHA_256 ラップアルゴリズムには OpenSSL バージョン 3.x 以降が必要です。

1. 256 ビット AES 対称暗号化キーを生成する

このコマンドは、256 ランダムビットで構成される AES 対称暗号化キーを生成し、aes-key.bin ファイルに保存します

```
# Generate a 32-byte AES symmetric encryption key
$ openssl rand -out aes-key.bin 32
```

2. AES 対称暗号化キーを使用してキーマテリアルを暗号化する

このコマンドは、AES 対称暗号化キーを使用してキーマテリアルを暗号化し、暗号化されたキーマテリアルを `key-material-wrapped.bin` ファイルに保存します。

このコマンド例では、

- *PlaintextKeyMaterial.bin* は、`PlaintextKeyMaterial.bin`、`HMAC_384_PlaintextKeyMaterial.bin`、インポートするキーマテリアルを含むファイルです。
- *aes-key.bin* は、前のコマンドで生成した 256 ビット AES 対称暗号化キーを含むファイルです。

```
# Encrypt your key material with the AES symmetric encryption key
$ openssl enc -id-aes256-wrap-pad \
  -K "$(xxd -p < aes-key.bin | tr -d '\n')" \
  -iv A65959A6 \
  -in PlaintextKeyMaterial.bin \
  -out key-material-wrapped.bin
```

3. AES 対称暗号化キーをパブリックキーで暗号化する

このコマンドは、ダウンロードしたパブリックキーと `RSAES_OAEP_SHA_256` ラップアルゴリズムで AES 対称暗号化キーを暗号化し、DER エンコードして `aes-key-wrapped.bin` ファイルに保存します。

このコマンド例では、

- *WrappingPublicKey.bin* は、ダウンロードしたパブリックラップキーを含むファイルです。コンソールからパブリックキーをダウンロードした場合、このファイルの名前は `wrappingKey_KMS_key_key_ID_timestamp` (例えば、`wrappingKey_f44c4e20-f83c-48f4-adc6-a1ef38829760_0809092909`) となります
- *aes-key.bin* は、このシーケンスの例の最初のコマンドで生成した 256 ビット AES 対称暗号化キーを含むファイルです。

```
# Encrypt your AES symmetric encryption key with the downloaded public key
$ openssl pkeyutl \
  -encrypt \
  -in aes-key.bin \
  -out aes-key-wrapped.bin \
```

```
-inkey WrappingPublicKey.bin \  
-keyform DER \  
-pubin \  
-pkeyopt rsa_padding_mode:oaep \  
-pkeyopt rsa_oaep_md:sha256 \  
-pkeyopt rsa_mgf1_md:sha256
```

4. インポートするファイルを生成する

暗号化されたキーマテリアルを含むファイルと暗号化された AES キーを含むファイルを連結します。これらを `EncryptedKeyMaterial.bin` ファイルに保存します。このファイルは、[ステップ 4: キーマテリアルのインポート](#) にインポートするファイルです。

このコマンド例では、

- *key-material-wrapped.bin* は、暗号化されたキーマテリアルを含むファイルです。
- *aes-key-wrapped.bin* は、暗号化された AES 暗号化キーを含むファイルです。

```
# Combine the encrypted AES key and encrypted key material in a file  
$ cat aes-key-wrapped.bin key-material-wrapped.bin > EncryptedKeyMaterial.bin
```

[ステップ 4: キーマテリアルのインポート](#) に進みます。

ステップ 4: キーマテリアルのインポート

[キーマテリアルを暗号化する](#)と、キーマテリアルをインポートして AWS KMS key で使用できます。キーマテリアルをインポートするには、[ステップ 3: キーマテリアルを暗号化する](#) から暗号化されたキーマテリアルと、[ステップ 2: ラップパブリックキーおよびインポートトークンのダウンロード](#) でダウンロードしたインポートトークンをアップロードします。[パブリックキーとインポートトークンをダウンロード](#)したときに指定したものと同一 KMS キーに、キーマテリアルをインポートする必要があります。キーマテリアルが正常にインポートされると、KMS キーの[キーステータス](#)が `Enabled` に変化し、暗号化オペレーションで KMS キーを使用できるようになります。

キーマテリアルをインポートする場合、キーマテリアルの[有効期限をオプションで設定](#)できます。キーマテリアルが有効期限切れになると、AWS KMS はキーマテリアルを削除し、KMS キーは使用不可能になります。キーマテリアルをインポートした後は、現在のインポートの有効期限を設定、変更、またはキャンセルできません。これらの値を変更するには、同じキーマテリアルを[再インポート](#)する必要があります。

EXTERNAL オリジンを持つすべての KMS キーでは、インポートされた最初のキーマテリアルが最新になり、永続的に関連付けられます。EXTERNAL オリジンを使用した単一リージョンの対称暗号化キーは、オンデマンドローテーションをサポートします。複数のキーマテリアルを、オンデマンドローテーションをサポートするインポートされたキーに関連付けることができます。新しいキーマテリアルを KMS キーに関連付けるには、[ImportKeyMaterial](#) アクション NEW_KEY_MATERIAL で importType パラメータを に設定する必要があります。このキーマテリアルは、[RotateKeyOnDemand](#) アクションを実行するまで、キーに永続的に関連付けられません。それまでは、このキーマテリアルは PENDING_ROTATION 状態です。オプションのパラメータのデフォルト値は ImportType です EXISTING_KEY_MATERIAL。ImportType パラメータを省略するか、として指定するときは EXISTING_KEY_MATERIAL、以前に KMS キーに関連付けられているキーマテリアルをインポートする必要があります。

EXTERNAL リージンを持つ非対称、HMAC、またはマルチリージョン KMS キーの場合、キーに関連付けることができるキーマテリアルは 1 つだけです。AWS KMS は、ImportType パラメータを使用して [ImportKeyMaterial](#) API リクエストを拒否します。

KMS キーに永続的に関連付けられているすべてのキーマテリアルがインポートされると、KMS キーは暗号化オペレーションで使用できます。これらのキーマテリアルのいずれかが削除されたり、有効期限が切れたりすると、KMS キーの状態は に変わり PendingImport、キーは暗号化オペレーションに使用できません。

キーマテリアルをインポートするには、[AWS KMS コンソール](#) または [ImportKeyMaterial](#) API を使用できます。HTTP リクエストを作成することにより、または [AWS SDKs](#)、[AWS Command Line Interface](#) または [AWS Tools for PowerShell](#) を使用することにより、直接 API を使用できます。

キーマテリアルをインポートすると、[ImportKeyMaterial エントリ](#) が AWS CloudTrail ログに追加され、ImportKeyMaterial オペレーションが記録されます。CloudTrail エントリは、AWS KMS コンソールと AWS KMS API のどちらを使用する場合でも同じです。

有効期限の設定 (オプション)

KMS キーのキーマテリアルをインポートするとき、オプションで、キーマテリアルの有効期限 (日付と時刻) をインポートした日から最大 365 日間までの範囲で設定できます。インポートされたキーマテリアルの有効期限が切れると、はそれ AWS KMS を削除します。このアクションにより KMS キーの [キーキステータス](#) は PendingImport に変更され、暗号化オペレーションで使用できなくなります。KMS キーを使用する場合は、[元のキーマテリアルを再度インポート](#) する必要があります。

インポートされたキーマテリアルが頻繁に期限切れになるようにすれば規制要件を満たすのには役立ちますが、KMS キーで暗号化されたデータのリスクが高まります。元のキーマテリアルのコピーを

再インポートするまでは、キーマテリアルの期限が切れた KMS キーは使用できず、KMS キーで暗号化されたデータにはアクセスできなくなります。元のキーマテリアルのコピーを紛失するなど何らかの理由でキーマテリアルを再インポートしなければ、その KMS キーは永久に使用できなくなり、その KMS キーで暗号化されたデータは回復不能となります。

このリスクを軽減するには、インポートされたキーマテリアルのコピーにアクセスできることを確認し、キーマテリアルが期限切れになって AWS ワークロードを中断する前に削除して再インポートするシステムを設計します。インポートされたキーマテリアルの、有効期限の[アラームを設定](#)し、期限切れになる前に、キーマテリアルを再インポートする十分な時間を確保しておくことが推奨されます。また、CloudTrail ログを使用して、[キーマテリアルをインポート \(および再インポート\)](#) したり [インポートしたキーマテリアルを削除](#) したりするオペレーションや、[期限切れのキーマテリアルを削除](#) する AWS KMS オペレーションを監査することもできます。

AWS KMS は、削除されたキーマテリアルを復元、復元、または再現することはできません。有効期限を設定する代わりに、インポートしたキーマテリアルをプログラムで定期的に[削除](#)し[再インポート](#)することができますが、元のキーマテリアルのコピーを保持するための要件は同じです。

キーマテリアルをインポートするときに、インポートしたキーマテリアルの有効期限が切れているかどうか、および、いつ切れるのかを判断します。ただし、有効期限のオンとオフを切り替えることも、キーマテリアルを再インポートして新しい有効期限を設定することもできます。有効期限を有効 (KEY_MATERIAL_EXPIRES) および無効 (KEY_MATERIAL_DOES_NOT_EXPIRE) にするときは [ImportKeyMaterial](#) の ExpirationModel パラメーターを使用し、有効期限を設定するときは ValidTo パラメータを使用します。最大日数はインポートした日から 365 日目です。最小日数はありませんが、時刻は未来の時刻でなければなりません。

キーマテリアルの説明を設定する

オEXTERNALリジンを持つ単一リージョンの対称暗号化キーには、複数のキーマテリアルを関連付けることができます。このようなキーにキーマテリアルをインポートするときに、オプションのキーマテリアルの説明を指定できます。説明を使用して、対応するキーマテリアルが外部で永続的に維持されている場所を追跡できます AWS KMS。

キーマテリアルの再インポート

キーマテリアルがインポートされた KMS キーを管理する場合、キーマテリアルを再インポートしなければならない場合があります。キーマテリアルを再インポートして、有効期限の切れるキーマテリアルまたは削除されたキーマテリアルを置き換えるか、キーマテリアルの有効期限モデルまたは有効期限を変更することもできます。

キーマテリアルは、スケジュールがセキュリティ要件を満たしていれば、いつでも再インポートできます。キーマテリアルが有効期限に達するかそれに近づくのを待つ必要はありません。

キーマテリアルを再インポートする手順は、キーマテリアルを初めてインポートする場合と同じですが、以下の例外があります。

- 新しい KMS キーを作成する代わりに、既存の KMS キーを使用します。インポートの手順の [ステップ 1](#) はスキップできます。
- キーマテリアルを再インポートするときに、有効期限モデルと有効期限を変更することができます。

KMS キーにキーマテリアルをインポートするたびに、KMS キーの [新しいラップキーおよびインポートトークンをダウンロードして使用する](#) 必要があります。このラッピングの手順は、キーマテリアルの内容には影響しません。そのため、異なるラッピングキーおよび異なるラッピングアルゴリズムを使用して同じキーマテリアルをインポートできます。

新しいキーマテリアルをインポートする

インポートされたキーマテリアルを使用して対称暗号化 KMS キーでオンデマンドローテーションを実行するには、最初にキーに以前関連付けられていなかった新しいキーマテリアルをインポートする必要があります。ImportType パラメータを に設定して [ImportKeyMaterial](#) オペレーション NEW_KEY_MATERIAL を使用して、このタスクを実行します。この方法でインポートされたキーマテリアルは、[RotateKeyOnDemand](#) オペレーションを実行するか、でキーをローテーションするまで PENDING_ROTATION 状態になります AWS Management Console。KMS キーは、いつでも最大 1 つのキーマテリアルを PENDING_ROTATION 状態にすることができます。

キーマテリアルのインポート (コンソール)

を使用してキーマテリアル AWS Management Console をインポートできます。

1. 「ラップされたキーマテリアルのアップロード」のページが表示されている場合は、[Step 8](#) に進んでください。
2. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
3. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
4. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
5. 公開キーとインポートトークンをダウンロードした KMS キーの、キー ID またはエイリアスを選択します。

6. [暗号構成] タブを選択し、その値を表示します。タブは、KMS キーの詳細ページの一般設定セクションにあります。

キーマテリアルは、EXTERNAL (キーマテリアルのインポート) の Origin を含む KMS キーにのみインポートできます。インポートされたキーマテリアルで KMS キーを作成する方法の詳細については、[キーの AWS KMS キーマテリアルのインポート](#) を参照してください。

7. 非対称キー、HMAC キー、マルチリージョンキーの場合は、キーマテリアルタブを選択し、キーマテリアルのインポートを選択します。単一リージョンの対称暗号化キーの場合は、キーマテリアルとローテーションタブを選択します。次に、初期キーマテリアルをインポートするか、新しいキーマテリアルをインポートするか、キーマテリアルを再インポートを選択します。キーマテリアルの再インポートオプションは、キーマテリアルテーブルのActionsメニューで使用できます。

キーマテリアルをダウンロードし、トークンをインポートし、キーマテリアルを暗号化した場合は、[次へ] を選択します。

8. [暗号化されたキーマテリアルとインポートトークン] セクションで、次の操作を行います。
 - a. [ラップされたキーマテリアル] で、[ファイルを選択] を選択します。次に、ラップされた (暗号化された) キーマテリアルを含むファイルをアップロードします。
 - b. [トークンのインポート] で、[ファイルを選択] を選択します。[ダウンロード](#)したインポートトークンを含むファイルをアップロードします。
9. [有効期限オプション] セクションで、キーマテリアルの有効期限が切れているかどうかを判断します。有効期限の日時を設定するには、[キーマテリアルの有効期限] を選択し、カレンダーを使用して日付と時刻を選択します。現在の日付から 365 日後を上限として、日付を指定できます。
10. 対称暗号化キーの場合、オプションでインポートするキーマテリアルの説明を指定できます。
11. キーマテリアルのインポート を選択します。

キーマテリアルのインポート (AWS KMS API)

キーマテリアルをインポートするには、[ImportKeyMaterial](#) オペレーションを使用します。次の例では [AWS CLI](#) を使用しますが、サポートされているすべてのプログラミング言語を使用できます。

この例を使用するには:

1. **1234abcd-12ab-34cd-56ef-1234567890ab** を、公開キーとインポートトークンをダウンロードしたときに指定した KMS キーのキー ID と置き換えます。KMS キーを識別するには、そ

の [キー ID](#) または [キー ARN](#) を使用します。このオペレーションに [エイリアス名](#) や [エイリアス ARN](#) を使用することはできません。

2. *EncryptedKeyMaterial.bin* を、暗号化されたキーマテリアルを含むファイル名に置き換えます。
3. *ImportToken.bin* を、インポートトークンを含むファイル名に置き換えます。
4. インポートしたキーマテリアルを有効期限切れにする場合は、expiration-model パラメータの値をデフォルトの KEY_MATERIAL_EXPIRES に変更するか、expiration-model パラメータを省略します。次に valid-to パラメータの値を、キーマテリアルを有効期限切れにする日時に置き換えます。リクエストの時点から 365 日後までの日時を設定できます。

```
$ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
  --import-token fileb://ImportToken.bin \
  --expiration-model KEY_MATERIAL_EXPIRES \
  --valid-to 2023-06-17T12:00:00-08:00
```

インポートしたキーマテリアルを有効期限切れにしない場合は、expiration-model パラメータを KEY_MATERIAL_DOES_NOT_EXPIRE に設定し、コマンドの valid-to パラメータを省略します。

```
$ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
  --import-token fileb://ImportToken.bin \
  --expiration-model KEY_MATERIAL_DOES_NOT_EXPIRE
```

5. KMS キーに関連付けられていない新しいキーマテリアルをインポートする場合は、ImportType パラメータを に設定します NEW_KEY_MATERIAL。このオプションは、単一リージョンの対称暗号化キーでのみ使用できます。これらのキーについては、オプションの KeyMaterialDescription パラメータを使用して、次のコマンドライン例でインポートされたキーマテリアルの説明を設定することもできます。

```
$ aws kms import-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --encrypted-key-material fileb://EncryptedKeyMaterial.bin \
  --import-token fileb://ImportToken.bin \
  --expiration-model KEY_MATERIAL_EXPIRES \
  --valid-to 2023-06-17T12:00:00-08:00 \
  --import-type NEW_KEY_MATERIAL \
  --key-material-description "Q2 2025 Rotation"
```

i Tip

コマンドが成功しない場合は、`KMSInvalidStateException` または `NotFoundException` が表示されることがあります。リクエストは再試行できます。

キーストアに KMS AWS CloudHSM キーを作成する

AWS CloudHSM キーストアを作成したら、キーストア AWS KMS keys で 作成できます。これらは、 が AWS KMS 生成するキーマテリアルを持つ [対称暗号化 KMS](#) キーである必要があります。カスタムキーストアで [非対称 KMS キー](#)、[HMAC KMS キー](#)、または [インポートされたキーマテリアル](#) を持つ KMS キーを作成することはできません。カスタムキーストア内の対称暗号化 KMS キーを使用して、非対称データキーペアを生成することもできません。KMS は IPv6 経由で AWS CloudHSM キーストアと通信できません。

AWS CloudHSM キーストアで KMS キーを作成するには、AWS CloudHSM キーストア [を関連する AWS CloudHSM クラスターに接続](#) し、クラスターに異なるアベイラビリティゾーンに少なくとも 2 つのアクティブな HSMs が含まれている必要があります。接続ステータスと HSM の数を確認するには、[AWS CloudHSM キーストアのページ](#) を AWS Management Console に表示します。API オペレーションを使用する場合は、[DescribeCustomKeyStores](#) オペレーションを使用して、AWS CloudHSM キーストアが接続されていることを確認します。クラスターとそのアベイラビリティゾーン内のアクティブな HSMs の数を確認するには、AWS CloudHSM [DescribeClusters](#) オペレーションを使用します。

AWS CloudHSM キーストアに KMS キーを作成すると、AWS KMS は KMS キーを作成します AWS KMS。ただし、関連付けられた AWS CloudHSM クラスターに KMS キーのキーマテリアルが作成されます。具体的には、 は作成した CU としてクラスターに AWS KMS サインインします。[kmsuser](#) 次に、クラスター内に永続的で抽出不可能な 256 ビットの Advanced Encryption Standard (AES) 対称キーが作成され、AWS KMS が [キーラベルの属性値](#) を設定します。これはクラスター内で、KMS キーの Amazon リソースネーム (ARN) にのみ表示されます。

コマンドが成功すると、新しい KMS キーの [キーステータス](#) は Enabled になり、そのオリジンは `AWS_CLOUDHSM` になります。作成後に KMS キーのオリジンを変更することはできません。AWS KMS コンソールのキーストアで、または [DescribeKey](#) オペレーションを使用して KMS AWS CloudHSM キーを表示すると、キー ID、キーの状態、作成日などの一般的なプロパティが表示され

ます。カスタムキーストア ID と AWS CloudHSM クラスター ID (オプション) を確認することもできます。

AWS CloudHSM キーストアで KMS キーを作成しようとしても失敗した場合は、エラーメッセージを使用して原因を特定してください。AWS CloudHSM キーストアが接続されていない (CustomKeyStoreInvalidStateException) か、関連付けられた AWS CloudHSM クラスターに、このオペレーションに必要な 2 つのアクティブな HSMs () がないことを示している可能性があります CloudHsmClusterInvalidConfigurationException。ヘルプについては、を参照してください [カスタムキーストアのトラブルシューティング](#)。

AWS CloudHSM キーストアに KMS キーを作成する オペレーションの AWS CloudTrail ログの例については、「」を参照してください [CreateKey](#)。

CloudHSM キーストアに新しい KMS キーを作成する

対称暗号化 KMS キーは、コンソールの AWS CloudHSM キーストアで AWS KMS 作成することも、[CreateKey](#) オペレーションを使用して作成することもできます。

AWS KMS コンソールの使用

キーストアに対称暗号化 KMS AWS CloudHSM キーを作成するには、次の手順に従います。

Note

エイリアス、説明、またはタグには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。
5. [対称] を選択します。
6. [Key usage] (キーの使用) では、[Encrypt and decrypt] (暗号化および復号化) オプションがすでに選択されています。この設定は変更しないでください。
7. [Advanced options (詳細オプション)] を選択します。

8. [キーマテリアルのオリジン] で、[AWS CloudHSM キーストア] を選択します。

キーストアでマルチリージョン AWS CloudHSM キーを作成することはできません。

9. [次へ] を選択します。
10. 新しい KMS AWS CloudHSM キーのキーストアを選択します。新しい AWS CloudHSM キーストアを作成するには、カスタムキーストアの作成を選択します。

選択した AWS CloudHSM キーストアのステータスは Connected である必要があります。関連付けられた AWS CloudHSM クラスターはアクティブで、異なるアベイラビリティゾーンに少なくとも 2 つのアクティブな HSMs が含まれている必要があります。

AWS CloudHSM キーストアの接続については、「」を参照してください[AWS CloudHSM キーストアを切断する](#)。HSM の追加については、AWS CloudHSM ユーザーガイドの [HSM の追加](#) を参照してください。

11. [次へ] を選択します。
12. KMS キーのエイリアスおよびオプションの説明を入力します。
13. (オプション)。[Add Tags] (タグの追加) ページで、KMS キーを識別または分類するタグを追加します。

AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、[のタグ AWS KMS](#) および [の ABAC AWS KMS](#) を参照してください。

14. [次へ] を選択します。
15. [Key administrators] (キー管理者) セクションで、KMS キーを管理できる IAM ユーザーとロールを選択します。詳細については、「[KMS キーの管理をキー管理者に許可する](#)」を参照してください。

メモ

IAM ポリシーでは、KMS キーを使用するアクセス許可を他の IAM ユーザーおよびロールに付与できます。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子のキーポリシーにキー管理者を追加します "Allow access for Key Administrators"。このステートメント識別子を変

更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

16. (オプション) これらのキー管理者がこの KMS キーを削除できないようにするには、ページの下部にある [Allow key administrators to delete this key] (キー管理者がこのキーを削除できるようにする) チェックボックスをオフにします。
17. [次へ] を選択します。
18. このアカウントセクションで、[暗号化オペレーション](#)で KMS キーを使用できる AWS アカウント IAM ユーザーとロールを選択します。詳細については、「[KMS キーの使用をキーユーザーに許可する](#)」を参照してください。

メモ

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

AWS KMS コンソールは、ステートメント識別子 "Allow use of the key" およびのキーポリシーにキーユーザーを追加します "Allow attachment of persistent resources"。これらのステートメント識別子を変更すると、コンソールでステートメントに加えた更新がどのように表示されるかに影響する可能性があります。

19. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある Other AWS アカウント セクションで、Add another AWS アカウントを選択し、外部アカウントの AWS アカウント ID を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

Note

他の の管理者は、ユーザーの IAM ポリシーを作成して KMS キーへのアクセスを許可 AWS アカウント する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

20. [Next] (次へ) を選択します。
21. キーのキーポリシーステートメントを確認します。キーポリシーを変更するには、編集 を選択します。
22. [次へ] を選択します。

23. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。
24. 終了したら、[Finish] (完了) を選択し、キーを作成します。

手順が成功すると、選択したキーストアに新しい KMS AWS CloudHSM キーが表示されます。新しい KMS キーの名前またはエイリアスを選択すると、詳細ページの暗号化設定タブに、KMS キーのオリジン (AWS CloudHSM)、カスタムキーストアの名前、ID、タイプ、AWS CloudHSM クラスターの ID が表示されます。手順が失敗すると、失敗を説明するエラーメッセージが表示されます。

Tip

カスタムキーストアで KMS キーをより簡単に識別できるようにするには、[Customer managed keys (カスタマーマネージドキー)] ページで、[Custom key store ID (カスタムキーストア ID)] 列を表示に追加します。右上隅にある歯車アイコンをクリックし、[Custom key store ID (カスタムキーストア ID)] を選択します。詳細については、「[コンソールの表示をカスタマイズする](#)」を参照してください。

AWS KMS API の使用

キーストアに AWS CloudHSM 新しい AWS KMS key (KMS キー) [CreateKey](#) オペレーションを使用します。CustomKeyStoreId パラメータを使用してカスタムキーストアを識別し、AWS_CLOUDHSM の Origin 値を指定します。

また、キーポリシーを指定するために Policy パラメータが必要になる場合もあります。キーポリシー ([PutKeyPolicy](#)) を変更したり、[説明](#) や [タグ](#) などのオプション要素をいつでも追加したりできます。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

次の例では、[DescribeCustomKeyStores](#) オペレーションを呼び出して、AWS CloudHSM キーストアが関連する AWS CloudHSM クラスターに接続されていることを確認します。デフォルトでは、このオペレーションは、アカウントとリージョンのすべてのカスタムキーストアを返します。特定の AWS CloudHSM キーストアのみを記述するには、その CustomKeyStoreId または CustomKeyStoreName パラメータを使用します (両方ではありません)。

このコマンドを実行する前に、例のカスタムキーストア ID を有効な ID に置き換えます。

Note

Description フィールドまたは Tags フィールドには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-1234567890abcdef0",
      "CustomKeyName": "ExampleKeyStore",
      "CustomKeyType": "AWS CloudHSM key store",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "CONNECTED"
    }
  ],
}
```

次のコマンド例では、[DescribeClusters](#) オペレーションを使用して、ExampleKeyStore (cluster-1a23b4cdefg) に関連付けられている AWS CloudHSM クラスターに少なくとも 2 つのアクティブな HSMs があることを確認します。クラスターにある HSM が 2 つに満たない場合、CreateKey オペレーションは失敗します。

```
$ aws cloudhsmv2 describe-clusters
{
  "Clusters": [
    {
      "SubnetMapping": {
        ...
      },
      "CreateTimestamp": 1507133412.351,
      "ClusterId": "cluster-1a23b4cdefg",
      "SecurityGroup": "sg-865af2fb",
      "HsmType": "hsm1.medium",
      "VpcId": "vpc-1a2b3c4d",
      "BackupPolicy": "DEFAULT",
      "Certificates": {
        "ClusterCertificate": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----\n"
      }
    }
  ]
}
```

```

    },
    "Hsms": [
      {
        "AvailabilityZone": "us-west-2a",
        "EniIp": "10.0.1.11",
        "ClusterId": "cluster-1a23b4cdefg",
        "EniId": "eni-ea8647e1",
        "StateMessage": "HSM created.",
        "SubnetId": "subnet-a6b10bd1",
        "HsmId": "hsm-abcdefghijkl",
        "State": "ACTIVE"
      },
      {
        "AvailabilityZone": "us-west-2b",
        "EniIp": "10.0.0.2",
        "ClusterId": "cluster-1a23b4cdefg",
        "EniId": "eni-ea8647e1",
        "StateMessage": "HSM created.",
        "SubnetId": "subnet-b6b10bd2",
        "HsmId": "hsm-zyxwvutsrq",
        "State": "ACTIVE"
      }
    ],
    "State": "ACTIVE"
  }
]
}

```

このコマンド例では、[CreateKey](#) オペレーションを使用して キーストアに KMS AWS CloudHSM キーを作成します。AWS CloudHSM キーストアで KMS キーを作成するには、キーストアのカスタム AWS CloudHSM キーストア ID を指定し、Originの値を指定する必要がありますAWS_CLOUDHSM。

レスポンスには、カスタムキーストアと AWS CloudHSM クラスターの IDs が含まれます。

このコマンドを実行する前に、例のカスタムキーストア ID を有効な ID に置き換えます。

```

$ aws kms create-key --origin AWS_CLOUDHSM --custom-key-store-id cks-1234567890abcdef0
{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",

```

```
"CreationDate": 1.499288695918E9,
"Description": "Example key",
"Enabled": true,
"MultiRegion": false,
"KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
"KeyManager": "CUSTOMER",
"KeyState": "Enabled",
"KeyUsage": "ENCRYPT_DECRYPT",
"Origin": "AWS_CLOUDHSM"
"CloudHsmClusterId": "cluster-1a23b4cdefg",
"CustomKeyStoreId": "cks-1234567890abcdef0"
"KeySpec": "SYMMETRIC_DEFAULT",
"CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
"EncryptionAlgorithms": [
  "SYMMETRIC_DEFAULT"
]
}
```

外部キーストアで KMS キーを作成する

外部キーストア [を作成して接続](#)したら、キーストア AWS KMS keys で 作成できます。これらは、[External key store] (外部キーストア) (EXTERNAL_KEY_STORE) のオリジン値をもつ [対称暗号化 KMS キー](#)である必要があります。カスタムキーストアで [非対称 KMS キー](#)、[HMAC KMS キー](#)、または [インポートされたキーマテリアル](#)を持つ KMS キーを作成することはできません。カスタムキーストア内の対称暗号化 KMS キーを使用して、非対称データキーペアを生成することもできません。

外部キーストアの KMS キーは AWS の外部にあるコンポーネントに依存するため、標準 KMS キーよりも遅延、耐久性、可用性に劣る可能性があります。外部キーストアで KMS キーを作成または使用する前に、外部キーストアプロパティをもつキーが必要であることを確認してください。

Note

一部の外部キーマネージャーは、外部キーストアに KMS キーを作成する簡単な方法を提供しています。詳細については外部キーマネージャーのドキュメントを参照してください。

外部キーストアで KMS キーを作成するには、以下を指定します。

- 外部キーストアの ID。

- 外部キーストア (EXTERNAL_KEY_STORE) の[キーマテリアルオリジン](#)。
- 外部キーストアに関連付けられた[外部キーマネージャー](#)内の既存の[外部キー](#) ID。この外部キーは、KMS キーのキーマテリアルとして機能します。KMS キーの作成後は、外部キー ID を変更できません。

AWS KMS は、暗号化および復号オペレーションのリクエストで外部キーストアプロキシに外部キー ID を提供します。AWS KMS は、外部キーマネージャーまたはその暗号化キーに直接アクセスすることはできません。

外部キーに加えて、外部キーストアの KMS キーにも AWS KMS キーマテリアルがあります。KMS キーで暗号化されたすべてのデータは、まずキーの AWS KMS キーマテリアル AWS KMS を使用して暗号化され、次に外部キーマネージャーによって外部キーを使用して暗号化されます。この[三重暗号化](#)プロセスにより、外部キーストアの KMS キーで保護された暗号文は、少なくとも AWS KMS のみで保護されている暗号文と同等の強度を持つことが保証されます。詳細については、「[外部キーストアの仕組み](#)」を参照してください。

CreateKey オペレーションが成功すると、新しい KMS キーの[キーステータス](#)は Enabled になります。[外部キーストアで KMS キーを表示する](#)と、キー ID、[キースペック](#)、[キーの使用法](#)、[キーの状態](#)、作成日などの一般的なプロパティが表示されます。また、外部キーストアの ID と[接続状態](#)、および外部キーの ID も表示されます。

外部キーストアで KMS キーの作成を試みて失敗した場合は、エラーメッセージを使用して原因を特定します。外部キーストアが接続されていない可能性 (CustomKeyStoreInvalidStateException)、外部キーストアプロキシが指定された外部キー ID (XksKeyNotFoundException) の外部キーを検出できない可能性、または外部キーが同じ外部キーストア XksKeyAlreadyInUseException 内の KMS キーにすでに関連付けられている可能性があります。

外部キーストアに KMS キーを作成する オペレーションの AWS CloudTrail ログの例については、「」を参照してください[CreateKey](#)。

トピック

- [外部キーストアの KMS キーの要件](#)
- [外部キーストアで新しい KMS キーを作成する](#)

外部キーストアの KMS キーの要件

外部キーストアに KMS キーを作成するには、外部キーストア、KMS キー、および KMS キーの外部暗号化キーマテリアルとなる外部キーの次のプロパティが必要です。

外部キーストアの要件

- 外部キーストアプロキシに接続する必要があります。

外部キーストアの[接続状態](#)を確認するには、「[外部キーストアを表示する](#)」を参照してください。
外部キーストアを接続するには、「[外部キーストアを接続および切断する](#)」を参照してください。

KMS キーの要件

KMS キーの作成後にこれらのプロパティを変更することはできません。

- キースペック: SYMMETRIC_DEFAULT
- キーの使用方法: ENCRYPT_DECRYPT
- キーマテリアルのオリジン: EXTERNAL_KEY_STORE
- マルチリージョン: FALSE

外部キーの要件

- 256 ビット AES 暗号化キー (256 ランダムビット)。外部キーの KeySpec は AES_256 である必要があります。
- 有効かつ使用可能です。外部キーの Status は ENABLED である必要があります。
- 暗号化および複合化用に設定されています。外部キーの KeyUsage には ENCRYPT と DECRYPT を含める必要があります。
- この KMS キーでのみ使用されます。外部キーストアの各 KMS key は、異なる外部キーに関連付けられている必要があります。

AWS KMS では、外部キーストア専用 to 外部キーを使用することも推奨しています。この制限により、キーに関する問題の特定と解決が容易になります。

- 外部キーストアの[外部キーストアプロキシ](#)からアクセスできます。

外部キーストアプロキシが指定された外部キー ID を使用してキーを検出できない場合、CreateKey オペレーションは失敗します。

- を使用して AWS のサービス 生成される予想されるトラフィックを処理できます。AWS KMS では、外部キーが 1 秒あたり最大 1800 リクエストを処理するように準備することをお勧めします。

外部キーストアで新しい KMS キーを作成する

AWS KMS コンソールの外部キーストアで、または [CreateKey](#) オペレーションを使用して、新しい KMS キーを作成できます。

AWS KMS コンソールの使用

外部キーストアで KMS キーを作成するには 2 つの方法があります。

- 方法 1 (推奨): 外部キーストアを選択し、その外部キーストアに KMS キーを作成します。
- 方法 2: KMS キーを作成し、それが外部キーストアにあることを示します。

キーを作成する前に外部キーストアを選択する方法 1 を使用する場合は、必要なすべての KMS キープロパティ AWS KMS を選択し、外部キーストアの ID を入力します。この方法により、KMS キーの作成時に発生する可能性のあるエラーを回避できます。

Note

エイリアス、説明、またはタグには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

方法 1 (推奨): 外部キーストアで起動する

この方法を使用するには、外部キーストアを選択し、KMS キーを作成します。AWS KMS コンソールは、必要なプロパティをすべて選択し、外部キーストアの ID を入力します。この方法により、KMS キーの作成時に発生する可能性のある多くのエラーを回避できます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[Custom key stores] (カスタムキーストア)、[External key stores] (外部キーストア) の順に選択します。
4. 外部キーストアの名前を選択します。

5. 右上隅にある [Create a KMS key in this key store] (このキーストアのKMS キーを作成する) を選択します。

外部キーストアが接続されていない場合は、接続するように求められます。接続に失敗した場合は、問題を解決し、外部キーストアに接続してから新しい KMS キーを作成する必要があります。

外部キーストアが接続されている場合は、キーを作成するための [Customer managed keys] (カスタマーマネージドキー) ページにリダイレクトされます。必要な [Key configuration] (キー設定) 値は既に選択されています。また、外部キーストアのカスタムキーストア ID も入力されていますが、変更可能です。

6. [外部キーマネージャー](#)に[外部キー](#)のキー ID を入力します。この外部キーは、KMS キーで使用するための[要件を満たしている](#)必要があります。キーの作成後にこの値を変更することはできません。

外部キーに複数の ID がある場合は、外部キーストアプロキシが外部キーの識別に使用するキー ID を入力します。

7. 指定された外部キーストアに KMS キーを作成する予定であることを確認します。
8. [次へ] を選択します。

この手順の残りの部分は、[標準の KMS キー作成](#)と同じです。

9. エイリアス (必須) と KMS キーの説明 (オプション) を入力します。
10. (オプション) [Add Tags] (タグの追加) ページで、KMS キーを識別または分類するタグを追加します。

AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、[のタグ AWS KMS](#) および [の ABAC AWS KMS](#) を参照してください。

11. [次へ] を選択します。
12. [Key administrators] (キー管理者) セクションで、KMS キーを管理できる IAM ユーザーとロールを選択します。詳細については、「[KMS キーの管理をキー管理者に許可する](#)」を参照してください。

 Note

IAM ポリシーでは、KMS キーを使用するアクセス許可を他の IAM ユーザーおよびロールに付与できます。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

13. (オプション) これらのキー管理者がこの KMS キーを削除できないようにするには、[Allow key administrators to delete this key] (キー管理者がこのキーを削除できるようにする) のチェックボックスをオフにします。

KMS キーの削除は破壊的で元に戻せないオペレーションであり、暗号文を回復不能にする可能性があります。外部キーマテリアルが存在しても、外部キーストアに対称 KMS キーを再作成することはできません。ただし、KMS キーを削除しても関連付けられた外部キーには影響しません。外部キーストアから KMS キーを削除する方法については、「[キーの削除に関する注意事項](#)」を参照してください。

14. [次へ] を選択します。

15. このアカウントセクションで、[暗号化オペレーション](#)で KMS キーを AWS アカウント 使用できる IAM ユーザーとロールを選択します。詳細については、「[KMS キーの使用をキーユーザーに許可する](#)」を参照してください。

Note

IAM ポリシーでは、KMS キーを使用するアクセス許可を他の IAM ユーザーおよびロールに付与できます。

IAM ベストプラクティスでは、長期の認証情報を持つ IAM ユーザーの使用は推奨されていません。可能な限り、一時的な認証情報を提供する IAM ロールを使用してください。詳細については、「IAM ユーザーガイド」の「[IAM でのセキュリティのベストプラクティス](#)」を参照してください。

16. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある Other AWS アカウント セクションで、Add another AWS アカウントを選択し、外部アカウントの AWS アカウント ID を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

Note

他の の管理者は、ユーザーの IAM ポリシーを作成して KMS キーへのアクセスを許可 AWS アカウント する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

17. [Next] (次へ) を選択します。
18. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。
19. 終了したら、[Finish] (完了) を選択し、キーを作成します。

方法 2: カスタマーマネージドキーから開始する

この手順は、AWS KMS キーマテリアルを使用して対称暗号化キーを作成する手順と同じです。ただし、この手順では、外部キーストアのカスタムキーストア ID と外部キーのキー ID を指定します。また、外部キーストアの KMS キーに[必要なプロパティ値](#) (キースペックやキーの使用法など) を指定する必要があります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. [Create key] (キーの作成) を選択します。
5. [対称] を選択します。
6. [Key usage] (キーの使用) では、[Encrypt and decrypt] (暗号化および復号化) オプションがすでに選択されています。この設定は変更しないでください。
7. [Advanced options (詳細オプション)] を選択します。
8. [Key material origin] (キーマテリアルのオリジン) で、[External key store] (外部キーストア) を選択します。
9. 指定された外部キーストアに KMS キーを作成する予定であることを確認します。
10. [次へ] を選択します。
11. 新しい KMS キーの外部キーストアを表す行を選択します。

接続されていない外部キーストアは選択できません。切断されたキーストアを接続するには、キーストア名を選択し、[Key store actions] (キーストアアクション) で [Connect] (接続) を選択します。詳細については、「[AWS KMS コンソールの使用](#)」を参照してください。

12. [外部キーマネージャーに外部キー](#)のキー ID を入力します。この外部キーは、KMS キーで使用するための[要件を満たしている](#)必要があります。キーの作成後にこの値を変更することはできません。

外部キーに複数の ID がある場合は、外部キーストアプロキシが外部キーの識別に使用するキー ID を入力します。

13. [次へ] を選択します。

この手順の残りの部分は、[標準の KMS キー作成](#)と同じです。

14. KMS キーのエイリアスおよびオプションの説明を入力します。
15. (オプション)。[Add Tags] (タグの追加) ページで、KMS キーを識別または分類するタグを追加します。

AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、[のタグ AWS KMS](#) および [の ABAC AWS KMS](#) を参照してください。

16. [次へ] を選択します。
17. [Key administrators] (キー管理者) セクションで、KMS キーを管理できる IAM ユーザーとロールを選択します。詳細については、「[KMS キーの管理をキー管理者に許可する](#)」を参照してください。

 Note

IAM ポリシーでは、KMS キーを使用するアクセス許可を他の IAM ユーザーおよびロールに付与できます。

18. (オプション) これらのキー管理者がこの KMS キーを削除できないようにするには、[Allow key administrators to delete this key] (キー管理者がこのキーを削除できるようにする) のチェックボックスをオフにします。

KMS キーの削除は破壊的で元に戻せないオペレーションであり、暗号文を回復不能にする可能性があります。外部キーマテリアルが存在しても、外部キーストアに对称 KMS キーを再作成することはできません。ただし、KMS キーを削除しても関連付けられた外部キーには影響しませ

ん。外部キーストアから KMS キーを削除する方法については、「[を削除する AWS KMS key](#)」を参照してください。

19. [次へ] を選択します。

20. このアカウントセクションで、[暗号化オペレーション](#)で KMS キーを AWS アカウント 使用できる IAM ユーザーとロールを選択します。詳細については、「[KMS キーの使用をキーユーザーに許可する](#)」を参照してください。

 Note

IAM ポリシーでは、KMS キーを使用するアクセス許可を他の IAM ユーザーおよびロールに付与できます。

21. (オプション) 暗号化オペレーションにこの KMS キーを使用すること AWS アカウント を他のユーザーに許可できます。これを行うには、ページの下部にある Other AWS アカウント セクションで、Add another AWS アカウントを選択し、外部アカウントの AWS アカウント ID を入力します。複数の外部アカウントを追加するには、この手順を繰り返します。

 Note

他の の管理者は、ユーザーの IAM ポリシーを作成して KMS キーへのアクセスを許可 AWS アカウント する必要があります。詳細については、「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。

22. [Next] (次へ) を選択します。

23. 選択したキー設定を確認します。戻って、すべての設定を変更することもできます。

24. 終了したら、[Finish] (完了) を選択し、キーを作成します。

手順が成功すると、選択した外部キーストアに新しい KMS キーが表示されます。新しい KMS キーの名前やエイリアスを選択すると、その詳細ページの [Cryptographic configuration] (暗号化設定) タブに、KMS キー ([External key store] (外部キーストア)) のオリジン、カスタムキーストアの名前、ID、タイプ、外部キーの ID、キー使用方法、ステータスが表示されます。手順が失敗すると、失敗を説明するエラーメッセージが表示されます。の場合は、「[外部キーストアのトラブルシューティング](#)」を参照してください。

i Tip

カスタムキーストアで KMS キーをより簡単に識別できるようにするには、[Customer managed keys] (カスタマーマネージドキー) ページで、[Origin] (オリジン) と [Custom key store ID] (カスタムキーストア ID) 列を表示に追加します。テーブルフィールドを変更するには、ページの右上隅にある歯車アイコンを選択します。詳細については、「[コンソールの表示をカスタマイズする](#)」を参照してください。

AWS KMS API の使用

外部キーストアで新しい KMS キーを作成するには、[CreateKey](#) オペレーションを使用します。以下のパラメータは必須です。

- Origin の値は EXTERNAL_KEY_STORE にする必要があります。
- CustomKeyId パラメータは外部キーストアを識別します。指定された外部キーストアの [ConnectionState](#) は、CONNECTED である必要があります。CustomKeyId と ConnectionState を検出するには、DescribeCustomKeyStores オペレーションを使用します。
- XksKeyId パラメータは外部キーを識別します。この外部キーは、KMS キーとの関連付けの[要件を満たしている](#)必要があります。

Policy または [タグ](#) パラメータを使用するなど、CreateKey オペレーションの任意のオプションパラメータを使用することもできます。

i Note

Description フィールドまたは Tags フィールドには、機密情報や重要情報を含めないでください。これらのフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

この例では、[CreateKey](#) オペレーションを使用して、外部キーストアで KMS キーを作成します。応答には、KMS キーのプロパティ、外部キーストアの ID、外部キーの ID、使用方法、ステータスが含まれます。

このコマンドを実行する前に、例のカスタムキーストア ID を有効な ID に置き換えます。

```
$ aws kms create-key --origin EXTERNAL_KEY_STORE --custom-key-store-  
id cks-1234567890abcdef0 --xks-key-id bb8562717f809024  
{  
  "KeyMetadata": {  
    "Arn": "arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
    "AWSAccountId": "111122223333",  
    "CreationDate": "2022-12-02T07:48:55-07:00",  
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",  
    "CustomKeyId": "cks-1234567890abcdef0",  
    "Description": "",  
    "Enabled": true,  
    "EncryptionAlgorithms": [  
      "SYMMETRIC_DEFAULT"  
    ],  
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",  
    "KeyManager": "CUSTOMER",  
    "KeySpec": "SYMMETRIC_DEFAULT",  
    "KeyState": "Enabled",  
    "KeyUsage": "ENCRYPT_DECRYPT",  
    "MultiRegion": false,  
    "Origin": "EXTERNAL_KEY_STORE",  
    "XksKeyConfiguration": {  
      "Id": "bb8562717f809024"  
    }  
  }  
}
```

キーの特定と表示

[AWS Management Console](#) または [AWS Key Management Service \(AWS KMS\) API](#) を使用して、管理する KMS キーや によって管理される KMS キーなど、各アカウントとリージョン AWS KMS keys で を表示できます AWS。

トピック

- [キー ID とキー ARN を検索する](#)
- [KMS キーの詳細にアクセスして一覧表示する](#)
- [さまざまなキータイプの特定](#)
- [コンソールの表示をカスタマイズする](#)
- [キーストアで KMS キーと AWS CloudHSM キーマテリアルを検索する](#)

キー ID とキー ARN を検索する

を識別するには AWS KMS key、[キー ID](#) または Amazon リソースネーム ([キー ARN](#)) を使用できます。[暗号化オペレーション](#)では、[エイリアス名](#)または[エイリアス ARN](#) を使用することもできます。

[AWS KMS コンソール](#)または [ListKeys](#) オペレーションを使用して、お使いのアカウントとリージョン内の各 KMS キーのキー ID とキー ARN を確認できます。

でサポートされている KMS キー識別子の詳細については AWS KMS、「」を参照してください[キー識別子 \(KeyId\)](#)。エイリアス名とエイリアス ARN を検索する方法については、「」を参照してください [KMS キーのエイリアス名とエイリアス ARN を見つける](#)。

AWS KMS コンソールの使用

1. <https://console.aws.amazon.com/kms> で AWS KMS コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ユーザーが作成および管理するアカウント内のキーを表示するには、ナビゲーションペインで [Customer managed keys] (カスタマーマネージドキー) を選択します。が AWS 作成および管理するアカウントのキーを表示するには、ナビゲーションペインでマネージドAWS キーを選択します。
4. KMS キーの[キー ID](#) を検索するには、KMS キーのエイリアスで始まる行を参照します。

デフォルトでは、[キー ID] 列がテーブルに表示されます。[キー ID] 列がテーブルに表示されない場合は、「[the section called “コンソールの表示をカスタマイズする”](#)」で説明されている手順に従って復元します。KMS キーの詳細ページで KMS キーのキー ID を表示することもできます。

Customer managed keys					Key actions ▼	Create key
					< 1 >	⚙
☐	Aliases ▲	Key ID ▼	Status	Creation date		
☐	key-test	1234abcd-12ab-34cd-56ef-1234567890ab	Enabled	Oct 19, 2018 12:43 PDT		

5. KMS キーの Amazon リソースネーム (ARN) を検索するには、キー ID またはエイリアスを選択します。[キー ARN](#) が [General configuration] セクションに表示されます。

General configuration		
Aliases key-test	Status Enabled	ARN arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
Description -	Creation date Nov 06, 2018 15:11 PST	

AWS KMS API の使用

の[キー ID](#) と[キー ARN](#) を検索するには AWS KMS key、[ListKeys](#) オペレーションを使用します。

ListKeys オペレーションは、呼び出し元のアカウントとリージョン内のすべての KMS キーのキー ID と Amazon リソースネーム (ARN) を返します。

例えば、ListKeys オペレーションに対するこの呼び出しでは、架空のアカウントの各 KMS キーの ID と ARN を返します。複数のプログラミング言語の例については、「[AWS SDK または CLI ListKeysで を使用する](#)」を参照してください。

```
$ aws kms list-keys
{
  "Keys": [
    {
```

```
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyArn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  {
    "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "KeyArn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
  }
]
```

KMS キーの詳細にアクセスして一覧表示する

[AWS KMS コンソール](#)または [DescribeKey](#) オペレーションを使用して、アカウントとリージョン内の KMS キーに関する詳細情報にアクセスして一覧表示できます。

以下の手順は、キー ID、キー仕様、キー用途などの KMS キーの詳細にアクセスする方法を示します。

AWS KMS コンソールの使用

各 KMS キーの詳細ページには、KMS キーのプロパティが表示されます。KMS キーの種類によって若干異なります。

KMS キーに関する詳細情報を表示するには、AWS マネージドキー またはカスターマネージドキーページで、KMS キーのエイリアスまたはキー ID を選択します。

KMS キーの詳細ページには、KMS キーのベーシックプロパティを表示する [General configuration] (一般設定) セクションがあります。また、キーポリシー、暗号化設定、タグ、キーマテリアルとローテーション (自動ローテーションまたはオンデマンドローテーションをサポートする KMS キーの場合)、リージョン性 (マルチリージョンキーの場合)、パブリックキー (非対称 KMS キーの場合) など、KMS キーのプロパティを表示および編集できるタブも含まれています。

Note

AWS KMS コンソールには、アカウントとリージョンで[表示するアクセス許可](#)を持つ KMS キーが表示されます。他の KMS キーは、表示、管理、使用するアクセス許可がある場合でも、コンソールに表示されません。他のアカウントの KMS キーを表示するには、[DescribeKey](#) オペレーションを使用します。

KMS キーのキーの詳細ページに移動するには

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ユーザーが作成および管理するアカウント内のキーを表示するには、ナビゲーションペインで [Customer managed keys] (カスタマーマネージドキー) を選択します。が AWS 作成および管理するアカウントのキーを表示するには、ナビゲーションペインでマネージドAWS キーを選択します。
4. キーの詳細ページを開くには、キーテーブルで KMS キーのキー ID またはエイリアスを選択します。

KMS キーに複数のエイリアスがある場合、エイリアスの概要 (+n 個追加) が、エイリアスの 1 つの名前の横に表示されます。エイリアスのサマリーを選択すると、キーの詳細ページの Aliases (エイリアス) タブを直接表示します。

≡ [KMS](#) > [Customer managed keys](#) > Key ID: 6f84178a-007d-4501-8229-abc6af9e609c

ⓘ | 🗨

6f84178a-007d-4501-8229-abc6af9e609c

Key actions ▼ Edit

General configuration

Alias

kms-test

ARN

arn:aws:kms:us-east-1:849743133991:key/6f84178a-007d-4501-8229-abc6af9e609c

Current key material ID

f2e38520f5b8424d0ccd2dc64530a25e830926738224b0cb97d8e62302549f81

Status

Enabled

Description

-

Creation date

May 26, 2025 01:49 PDT

Regionality

Single Region

Key policy

Cryptographic configuration

Tags

Key material and rotations

Aliases

Cryptographic configuration

Key Type

Symmetric

Origin

AWS KMS

Key Spec ⓘ

SYMMETRIC_DEFAULT

Key Usage

Encrypt and decrypt

次のリストでは、タブ内のフィールドを含め、詳細表示のフィールドについて説明します。これらのフィールドの一部は、テーブル表示の列としても使用できます。

エイリアス

場所: [Aliases] (エイリアス) タブ

KMS キーのわかりやすい名前。エイリアスを使用して、コンソールと一部の AWS KMS APIs で KMS キーを識別できます。詳細については、「[のエイリアス AWS KMS](#)」を参照してください。

エイリアスタブには、AWS アカウント および リージョンの KMS キーに関連付けられたすべてのエイリアスが表示されます。

ARN

場所: [General configuration] (一般設定) セクション

KMS キーの Amazon リソースネーム (ARN)。この値は KMS キーを一意に識別します。この値を使用して、AWS KMS API オペレーションで KMS キーを識別できます。

接続状態

[カスタムキーストア](#)がバックアップキーストアに接続しているかどうかを示します。このフィールドは、KMS キーがカスタムキーストアで作成された場合にのみ表示されます。

このフィールドの値の詳細については、「AWS KMS API リファレンスの「[ConnectionState](#)」を参照してください。

作成日

場所: [General configuration] (一般設定) セクション

KMS キーが作成された日時。この値は、デバイスの現地時間で表示されます。タイムゾーンはリージョンに依存しません。

有効期限切れとは異なり、作成時は KMS キーのみを参照し、キーマテリアルは参照しません。

CloudHSM クラスター ID

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーのキーマテリアルを含む AWS CloudHSM クラスターのクラスター ID。このフィールドは、KMS キーが[カスタムキーストア](#)で作成された場合にのみ表示されます。

CloudHSM クラスター ID を選択すると、AWS CloudHSM コンソールでクラスターページが開きます。

現在のキーマテリアル

場所: [General configuration] (一般設定) セクション

AWS_KMS オリジンを持つ対称暗号化キーは、自動ローテーションとオンデマンドローテーションの両方をサポートします。EXTERNAL オリジンを使用した単一リージョンの対称暗号化キーは、オンデマンドローテーションをサポートします。これらのキーには、キーに関連付けられた複数のキーマテリアルを含めることができます。最近ローテーションされたキーマテリアルは、暗号化と復号の両方に使用できます。このキーマテリアルは、現在のキーマテリアルとして識別されます。他のキーマテリアルは復号にのみ使用できます。KMS キーの自動キーローテーションまたはオンデマンドキーローテーションは、現在のキーマテリアルを変更します。

カスタムキーストア ID

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーを含む[カスタムキーストア](#)の ID。このフィールドは、KMS キーがカスタムキーストアで作成された場合にのみ表示されます。

カスタムキーストア ID を選択すると、AWS KMS コンソールでカスタムキーストアページが開きます。

カスタムキーストア名

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーを含む[カスタムキーストア](#)の名前。このフィールドは、KMS キーがカスタムキーストアで作成された場合にのみ表示されます。

カスタムキーストアのタイプ

場所: [Cryptographic configuration] (暗号化設定) タブ

カスタムキーストアが [AWS CloudHSM キーストア](#) と [外部キーストア](#) のいずれであるのかを示します。このフィールドは、KMS キーが[カスタムキーストア](#)で作成された場合にのみ表示されます。

説明

場所: [General configuration] (一般設定) セクション

書き込みおよび編集できる KMS キーの簡単な説明 (オプション)。カスタマーマネージドキーの説明を追加または更新するには、上記の一般設定で編集を選択します。

暗号化アルゴリズム

場所: [Cryptographic configuration] (暗号化設定) タブ

AWS KMSで KMS キーとともに使用できる暗号化アルゴリズムを一覧表示します。このフィールドは、[Key type] (キーのタイプ) が [Asymmetric] (非対称) で、[Key usage] (キーの用途) が [Encrypt and decrypt] (暗号化と復号) の場合にのみ表示されます。が AWS KMS サポートする暗号化アルゴリズムの詳細については、[SYMMETRIC_DEFAULT キー仕様](#)「」および「」を参照してください[暗号化および復号の RSA キー仕様](#)。

有効期限日

場所: [キーマテリアル] タブ

KMS キーのキーマテリアルの有効期限が切れる日時。このフィールドは、[インポートされたキーマテリアル](#)を持つ KMS キーに対してのみ表示されます。つまり、[Origin] (オリジン) が [External] (外部) で、KMS キーに有効期限付きキーマテリアルがある場合です。単一リージョンの対称暗号化キーには、複数のキーマテリアルを関連付けることができます。このようなキーの場合、このフィールドには、関連するキーマテリアルのいずれかの有効期限が切れる最も早い日時が表示されます。

外部キー ID

場所: [Cryptographic configuration] (暗号化設定) タブ

[外部キーストア](#)の KMS キーに関連付けられている[外部キー](#)のID。このフィールドは、外部キーストアの KMS キーにのみ表示されます。

外部キーのステータス

場所: [Cryptographic configuration] (暗号化設定) タブ

[外部キーストアプロキシ](#)が KMS キーに関連付けられた[外部キー](#)について報告した最新のステータス。このフィールドは、外部キーストアの KMS キーにのみ表示されます。

外部キーの使用

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーに関連付けられた[外部キー](#)で有効になっている、暗号化のオペレーション。このフィールドは、外部キーストアの KMS キーにのみ表示されます。

キーポリシー

場所: [Key policy] (キーポリシー) タブ

[IAM ポリシー](#)および[グラント](#)とともに KMS キーへのアクセスを制御します。KMS キーそれぞれに 1 つのキーポリシーがあります。これは、唯一の必須認可要素です。カスタマーマネージド

キーのキーポリシーを変更するには、キーポリシータブで編集を選択します。詳細については、「[the section called “キーポリシー”](#)」を参照してください。

キーマテリアルとローテーション

場所: キーマテリアルとローテーションタブ

このタブは、AWS_KMSオリジンを持つ対称暗号化キー (自動ローテーションとオンデマンドローテーションの両方をサポート) と、EXTERNALオリジンを持つ単一リージョンの対称暗号化キー (オンデマンドローテーションをサポート) に対してのみ表示されます。

タブには 3 つのパネルがあります。

自動ローテーション: [カスタマーマネージド KMS](#) キーのキーマテリアルの[自動ローテーション](#)を有効または無効にします。[カスタマーマネージド](#)キーのキーローテーションステータスを変更するには、チェックボックスを使用します。[AWS マネージドキー](#)のキーマテリアルのローテーションを有効または無効にすることはできません。AWS マネージドキー は毎年自動的にローテーションされます。

オンデマンドローテーション: [カスタマーマネージド](#)キーでキーマテリアルの[オンデマンドローテーション](#)を開始します。インポートされたキーの場合、今すぐローテーションオプションを使用するには、インポートされたキーマテリアルが既に PENDING_ROTATION状態になっている必要があります。

キーマテリアル: KMS キーに関連付けられているすべてのキーマテリアルを一覧表示します。各キーマテリアルには一意の識別子があり、その行には、キーマテリアルが KMS で使用できるようになったローテーション日など、キーマテリアルに関する追加情報が表示されます。インポートされたキーの場合、各行には、特定のキーマテリアルを削除したり、KMS キーに再インポートしたりするために使用できるアクションメニューもあります。

キー仕様

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーのキーマテリアルのタイプ。は、対称暗号化 KMS キー (SYMMETRIC_DEFAULT)、異なる長さの HMAC KMS キー、異なる長さの RSA キーの KMS キー、および異なる曲線の楕円曲線キー AWS KMS をサポートします。詳細については、「[Key spec](#)」を参照してください。

キーのタイプ

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーが対称か非対称かを示します。

キーの用途

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーを [Encrypt and decrypt] (暗号化および復号)、[Sign and verify] (署名および検証)、または [Generate and verify MAC] (MAC の生成と検証) のどれに使用できるかを示します。詳細については、「[Key usage](#)」を参照してください。

オリジン

場所: [Cryptographic configuration] (暗号化設定) タブ

KMS キーのキーマテリアルのソース。次の値を指定できます:

- AWS KMS が生成するキーマテリアル用の AWS KMS
- [AWS CloudHSM キーストア](#) の KMS キー用の AWS CloudHSM
- [インポートされたキーマテリアル](#) の外部 (BYOK)
- [外部キーストア](#) の KMS キー用の外部キーストア

MAC アルゴリズム

場所: [Cryptographic configuration] (暗号化設定) タブ

AWS KMS で HMAC KMS キーと使用できる MAC アルゴリズムのリストです。このフィールドは、[Key spec] (キーの仕様) が HMAC キー仕様 (HMAC_*) である場合にのみ表示されます。AWS KMS がサポートする MAC アルゴリズムについては、「[HMAC KMS キーの主な仕様](#)」を参照してください。

プライマリキー

場所: [リージョナリティ] タブ

この KMS キーが [マルチリージョンのプライマリキー](#) であることを示します。認可されたユーザーはこのセクションを使用して、別の関連するマルチリージョンキーに [プライマリキーを変更](#) できます。このフィールドは、KMS キーがマルチリージョンのプライマリキーである場合にのみ表示されます。

パブリックキー

場所: [Public key] (パブリックキー) タブ

非対称 KMS キーのパブリックキーを表示します。承認されたユーザーは、このタブを使用して [パブリックキーをコピーおよびダウンロード](#) できます。

リージョナリティ

場所: 一般設定セクションおよびリージョナリティータブ

KMS キーが単一リージョンキー、[マルチリージョンのプライマリキー](#)、または[マルチリージョンのレプリカキー](#)のいずれであるかを示します。このフィールドは、KMS キーがマルチリージョンキーである場合にのみ表示されます。

関連するマルチリージョンキー

場所: [リージョナリティ] タブ

関連するすべての[マルチリージョンのプライマリキーとレプリカキー](#) (現在の KMS キーを除く) を表示します。このフィールドは、KMS キーがマルチリージョンキーである場合にのみ表示されます。

プライマリキーの [Related multi-Region keys] (関連するマルチリージョンキー) セクションでは、承認されたユーザーが[新しいレプリカキーを作成](#)できます。

レプリカキー

場所: [リージョナリティ] タブ

この KMS キーが[マルチリージョンのレプリカキー](#)であることを示します。このフィールドは、KMS キーがマルチリージョンのレプリカキーである場合にのみ表示されます。

署名アルゴリズム

場所: [Cryptographic configuration] (暗号化設定) タブ

AWS KMSで KMS キーとともに使用できる署名アルゴリズムを一覧表示します。このフィールドは、[Key type] (キーのタイプ) が [Asymmetric] (非対称) で、[Key usage] (キーの用途) が [Sign and verify] (署名と検証) の場合にのみ表示されます。が AWS KMS サポートする署名アルゴリズムの詳細については、[署名および検証用の RSA キー仕様](#)「」および「」を参照してください。[楕円曲線のキー仕様](#)。

ステータス

場所: [General configuration] (一般設定) セクション

KMS キーのキーステータス。KMS キーはステータスが有効の場合にのみ、[暗号化オペレーション](#)で使用できます。各 KMS キーステータスと KMS キーで実行されるオペレーションへの影響の詳細については、[キーの AWS KMS キーステータス](#) を参照してください。

タグ

場所: [Tags] (タグ) タブ

KMS キーを記述するオプションのキーバリューペア。KMS キーのタグを追加または変更するには、[Tags] (タグ) タブで [Edit] (編集) を選択します。

AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。タグは、KMS キーへのアクセスの制御にも使用できます。KMS キーのタグ付けについては、[のタグ AWS KMS](#) および [の ABAC AWS KMS](#) を参照してください。

AWS KMS API の使用

[DescribeKey](#) オペレーションは、指定された KMS キーの詳細を返します。KMS キーを識別するには、その[キー ID](#)、[キー ARN](#)、[エイリアス名](#)、[エイリアス ARN](#) を使用します。

発信者のアカウントとリージョンの KMS キーのみが表示される [ListKeys](#) オペレーションとは異なり、許可されたユーザーは、DescribeKey オペレーションを使用して、他のアカウントの KMS キーに関する詳細を取得できます。

Note

DescribeKey レスポンスは、同じ値を持つ KeySpec および CustomerMasterKeySpec メンバーの両方を含みます。CustomerMasterKeySpec メンバーは非推奨です。

例えば、DescribeKey に対するこの呼び出しは、対称暗号化 KMS キーに関する情報を返します。レスポンスのフィールドは、[AWS KMS key の仕様](#)、[キーの状態](#)、[キーマテリアルのオリジン](#)によって異なります。複数のプログラミング言語の例については、「[AWS SDK または CLI DescribeKeyで を使用する](#)」を参照してください。

```
$ aws kms describe-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "Enabled": true,
```



```

    "KeySpec": "SYMMETRIC_DEFAULT",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "CreationDate": 1499988169.234,
    "MultiRegion": false,
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
    ],
    "CurrentKeyMaterialId":
    "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  }
}

```

この例では、署名と検証に使用される非対称 KMS キーで DescribeKey オペレーションを呼び出します。レスポンスには、この KMS キーに対して AWS KMS がサポートする署名アルゴリズムが含まれます。

```

$ aws kms describe-key --key-id 0987dcba-09fe-87dc-65ba-ab0987654321

{
  "KeyMetadata": {
    "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "Origin": "AWS_KMS",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
    "KeyState": "Enabled",
    "KeyUsage": "SIGN_VERIFY",
    "CreationDate": 1569973196.214,
    "Description": "",
    "KeySpec": "ECC_NIST_P521",
    "CustomerMasterKeySpec": "ECC_NIST_P521",
    "AWSAccountId": "111122223333",
    "Enabled": true,
    "MultiRegion": false,
    "KeyManager": "CUSTOMER",
    "SigningAlgorithms": [
        "ECDSA_SHA_512"
    ]
  }
}

```

```
}
```

さまざまなキータイプの特定

以下のトピックでは、AWS KMS コンソールおよび [DescribeKey](#) レスポンスでさまざまなキータイプを識別する方法について説明します。

KMS キーの詳細ページにある [暗号化設定] タブに移動する方法については、「[the section called “KMS キーの詳細にアクセスして一覧表示する”](#)」を参照してください。

トピック

- [非対称 KMS キーを特定する](#)
- [HMAC KMS キーの特定](#)
- [マルチリージョン KMS キーの特定](#)
- [インポートされたキーマテリアルを持つ KMS キーを特定する](#)
- [キーストア内の KMS AWS CloudHSM キーを特定する](#)
- [外部キーストアの KMS キーを特定する](#)

非対称 KMS キーを特定する

AWS KMS コンソールで

[カスターマネージドキー] テーブルの [キータイプ] 列は、各 KMS キーが対称であるか非対称であることを示します。[キータイプ] 値でテーブルをフィルタリングして、非対称 KMS キーのみを表示させることができます。詳細については、「[the section called “KMS キーのソートおよびフィルタリング”](#)」を参照してください。

KMS キーの詳細ページの [暗号化設定] タブに [キータイプ] が表示され、そのキーが対称か非対称かを示します。また、[キーの用途] も表示されます。これは、その非対称 KMS キーが暗号化と復号化、署名と検証、および共有シークレットの取得のうちどの用途に使用されるのかを示します。

DescribeKey レスポンス内で

非対称 KMS キーで `DescribeKey` オペレーションを呼び出すと、レスポンスには `KeySpec` と `KeyUsage` の値が含まれます。これらの値を使用して、KMS キーが対称か非対称かを確認できます。

KeySpec 値が SYMMETRIC_DEFAULT の場合、そのキーは対称暗号化 KMS キーです。非対称キー仕様の詳細については、「[キー仕様のリファレンス](#)」を参照してください。

KeyUsage 値が SIGN_VERIFY または KEY_AGREEMENT の場合、そのキーは非対称 KMS キーです。

DescribeKey オペレーションは、非対称 KMS キーに関する以下の詳細も返します。

- KeyUsage 値が ENCRYPT_DECRYPT である非対称 KMS キーの場合、オペレーションはそのキーの有効な暗号化アルゴリズムを一覧表示する EncryptionAlgorithms を返します。
- KeyUsage 値が SIGN_VERIFY である非対称 KMS キーの場合、オペレーションはそのキーの有効な署名アルゴリズムを一覧表示する SigningAlgorithms を返します。
- KeyUsage 値が KEY_AGREEMENT である非対称 KMS キーの場合、オペレーションはそのキーの有効なキーアグリーメントアルゴリズムを一覧表示する KeyAgreementAlgorithms を返します。

非対称 KMS キーの詳細については、「[the section called “非対称キー”](#)」を参照してください。

HMAC KMS キーの特定

AWS KMS コンソールで

HMAC KMS キーは [カスタマーマネージドキー] テーブルに含まれていますが、HMAC キーを識別するキー仕様値またはキー用途値でこのテーブルをソートまたはフィルタリングすることはできません。HMAC キーを見つけやすくするには、それらに固有のエイリアスまたはタグを割り当てます。そうすることで、エイリアスまたはタグによるソートまたはフィルタリングが可能になります。

KMS キーの詳細ページの [暗号化設定] タブに [キータイプ] が表示され、そのキーが対称か非対称かを示します。HMAC KMS キーは対称です。[暗号化設定] タブには、[キーの用途] も表示されます。HMAC KMS キーにおいて、有効なキー用途値は常に [MAC の生成と検証] となります。

DescribeKey レスポンス内で

HMAC KMS キーで DescribeKey オペレーションを呼び出すと、レスポンスには KeySpec と KeyUsage の値が含まれます。HMAC KMS キーの場合、キー用途値は常に GENERATE_VERIFY_MAC であり、キー仕様値は常に HMAC_ で始まります。

HMAC KMS キーの詳細については、「[the section called “HMAC キー”](#)」を参照してください。

マルチリージョン KMS キーの特定

AWS KMS コンソールで

[カスターマネージドキー] テーブルには、選択されているリージョンの KMS キーのみが表示されます。選択したリージョンでマルチリージョンのプライマリキーおよびレプリカキーを表示できます。AWS リージョンを変更するには、コンソールの右上隅にあるリージョンセクターを使用します。

[カスターマネージドキー] テーブルでマルチリージョンキーを識別しやすくするには、テーブルに [リージョンナリティ] を追加します。ヘルプについては、「[the section called “KMS キーテーブルのカスタマイズ”](#)」を参照してください。

マルチリージョン KMS キーの詳細ページには、[リージョンナリティ] タブがあります。プライマリキーの [リージョンナリティ] タブには、プライマリリージョンの変更ボタンと新しいレプリカキーの作成ボタンがあります。(レプリカキーのリージョンナリティタブには、どちらのボタンもありません)。関連するマルチリージョンキーセクションには、現在のキーに関連するすべてのマルチリージョンキーが一覧表示されます。現在のキーがレプリカキーの場合、このリストにはプライマリキーが含まれます。

関連するマルチリージョンキーテーブルから関連するマルチリージョンキーを選択すると、AWS KMS コンソールは選択したキーのリージョンに変更され、キーの詳細ページが開きます。例えば、以下の「関連するマルチリージョンキー」セクションの例から sa-east-1 リージョンのレプリカキーを選択すると、AWS KMS コンソールは sa-east-1 リージョンに変更され、そのレプリカキーの詳細ページが表示されます。レプリカキーのエイリアスまたはキーポリシーを表示するにはこのオペレーションを行います。リージョンを再度変更するには、ページの右上隅にあるリージョンセクターを使用します。

DescribeKey レスポンス内で

デフォルトでは、AWS KMS API オペレーションはリージョン別であり、現在または指定されたリージョンのリソースのみを返します。ただし、マルチリージョン KMS キーで DescribeKey オペレーションを呼び出すと、レスポンスには MultiRegionConfiguration 要素の他の AWS リージョンに関連するすべてのマルチリージョンキーが含まれます。

マルチリージョン KMS キーの詳細については、「[the section called “マルチリージョンキー”](#)」を参照してください。

インポートされたキーマテリアルを持つ KMS キーを特定する

AWS KMS コンソールで

インポートされたキーマテリアルを持つ KMS キーを [カスターマネージドキー] テーブルで識別しやすくするため、テーブルに [オリジン] 列を追加します。[オリジン] 列によって、EXTERNAL (キーマテリアルのインポート) のオリジンプロパティ値を持つ KMS キーを簡単に識別できます。ヘルプについては、「[the section called “KMS キーテーブルのカスタマイズ”](#)」を参照してください。

KMS キーの詳細ページにある [暗号化設定] タブには、その KMS キーのキーマテリアルのソースを示す [オリジン] が表示されます。インポートされたキーマテリアルを持つ KMS キーの場合、オリジン値は常に [外部] (インポートキーマテリアル) となります。詳細ページには、インポートされたキーマテリアルに関する詳細情報を提供する [キーマテリアル] タブも含まれます。EXTERNAL オリジンを持つ単一リージョンの対称暗号化キーは、オンデマンドローテーションをサポートし、複数のキーマテリアルを関連付けることができます。このようなキーの場合、タブにはキーマテリアルとローテーションというラベルが付けられます。

DescribeKey レスポンス内で

インポートされたキーマテリアルを持つ KMS キーで DescribeKey オペレーションを呼び出すと、レスポンスには Origin、ExpirationModel、ValidTo の各値が含まれます。インポートされたキーマテリアルを持つ KMS キーの場合、オリジン値は常に EXTERNAL となります。ExpirationModel 値は、そのキーマテリアルに有効期限が設定されているかどうかを示し、ValidTo 値はキーマテリアルの有効期限を示します。複数のキーマテリアルがキーに関連付けられている場合、ValidTo 値はすべてのキーマテリアル (保留中の 1 つのローテーションを除く) の最も早い有効期限を示し ExpirationModel、これらのキーマテリアルのいずれも期限切れに設定されていない DOES_NOT_EXPIRE 場合にのみ に設定されます。詳細については、「[有効期限の設定 \(オプション\)](#)」を参照してください。

インポートされたキーマテリアルを持つ KMS キーの詳細については、「[the section called “インポートされたキーマテリアル”](#)」を参照してください。

キーストア内の KMS AWS CloudHSM キーを特定する

AWS KMS コンソールで

カスターマネージドキーテーブルの AWS CloudHSM キーストアで KMS キーを識別しやすくするには、オリジン列をテーブルに追加します。オリジン列では、AWS CloudHSM のオリジン

プロパティ値を持つ KMS キーを簡単に識別できます。ヘルプについては、「[the section called “KMS キーテーブルのカスタマイズ”](#)」を参照してください。

KMS キーの詳細ページにある [暗号化設定] タブには、その KMS キーのキーマテリアルのソースを示す [オリジン] が表示されます。キーストアの KMS AWS CloudHSM キーの場合、オリジン値は常に `aws:cloudhsm` です。

AWS CloudHSM キーストアの KMS キーの場合、暗号化設定タブには、KMS キーに関連付けられた AWS CloudHSM キーストアと AWS CloudHSM クラスターに関する情報を提供するカスタムキーストアというセクションが追加されています。

DescribeKey レスポンス内で

AWS CloudHSM キーストアの KMS キーで DescribeKey オペレーションを呼び出すと、レスポンスにはキーマテリアルのソースを示す `Origin` が含まれます。キーストアの KMS AWS CloudHSM キーの場合、オリジン値は常に `aws:cloudhsm` です。オペレーションは、AWS CloudHSM キーストアの KMS キーに対して次の特別なフィールドも返します。

- `CloudHsmClusterId`
- `CustomKeyId`

AWS CloudHSM キーストアの詳細については、「[the section called “AWS CloudHSM キーストア”](#)」を参照してください。

外部キーストアの KMS キーを特定する

AWS KMS コンソールで

外部キーストアの KMS キーを [カスタマーマネージドキー] テーブルで識別しやすくするには、[オリジン] 列をテーブルに追加します。[オリジン] 列により、[外部キーストア] のオリジンプロパティ値を持つ KMS キーを一目で識別できます。ヘルプについては、「[the section called “KMS キーテーブルのカスタマイズ”](#)」を参照してください。

KMS キーの詳細ページにある [暗号化設定] タブには、その KMS キーのキーマテリアルのソースを示す [オリジン] が表示されます。外部キーストアの KMS キー場合、オリジン値は常に [外部キーストア] となります。

外部キーストアの KMS キーの場合、[暗号化設定] タブには [カスタムキーストア] と [外部キー] の 2 つの追加セクションが含まれます。[カスタムキーストア] テーブルは、その KMS キーに関

連付けられた外部キーストアに関する情報を提供します。[外部キーストア] テーブルは、外部キーストアの KMS キーの AWS KMS コンソールにのみ表示されます。このセクションは、KMS キーに関連付けられている外部キーの情報を表示します。[外部キー](#)は、外部キーストアの KMS キーのキーマテリアル AWS として機能する 外の暗号化キーです。KMS キーを使用して暗号化または復号する際、オペレーションは、指定された外部キーを使用し、[外部キーマネージャー](#)によって実行されます。

[External key] (外部キー) セクションには、次の値が表示されます。

外部キー ID

外部キーマネージャーの外部キーの識別子です。これは、外部キーストアプロキシが外部キーを識別するために使用する値です。外部キー ID は KMS キーの作成時に指定します。この ID を変更することはできません。KMS キーの作成に使用した外部キー ID の値が変更または無効になった場合は、[KMS キーを削除するようにスケジュールして](#)、正しい外部キー ID 値を使用し、[新しい KMS キーを作成する](#)必要があります。

DescribeKey レスポンス内で

外部キーストアの KMS キーで DescribeKey オペレーションを呼び出すと、レスポンスにはキーマテリアルのソースを示す Origin が含まれます。キーストアの KMS AWS CloudHSM キーの場合、オリジン値は常に EXTERNAL_KEY_STORE です。オペレーションは、その KMS キーに関連付けられた外部キーストアを特定する CustomKeyId 要素も返します。

外部キーストアの使用の詳細については、「[the section called “外部キーストア”](#)」を参照してください。

コンソールの表示をカスタマイズする

AWS KMS コンソールの表示をカスタマイズして、KMS キーを見つけやすくすることができます。[AWS マネージドキー] および [カスタマー マネージドキー] ページに表示されるテーブルをカスタマイズして、最も必要な情報を見やすく表示したり、テーブル内に返される KMS キーをソートしてフィルタリングします。

トピック

- [KMS キーのソートおよびフィルタリング](#)
- [KMS キーテーブルのカスタマイズ](#)

KMS キーのソートおよびフィルタリング

コンソールで KMS キーを検索しやすくするために、キーテーブルをソートしてフィルタリングできます。

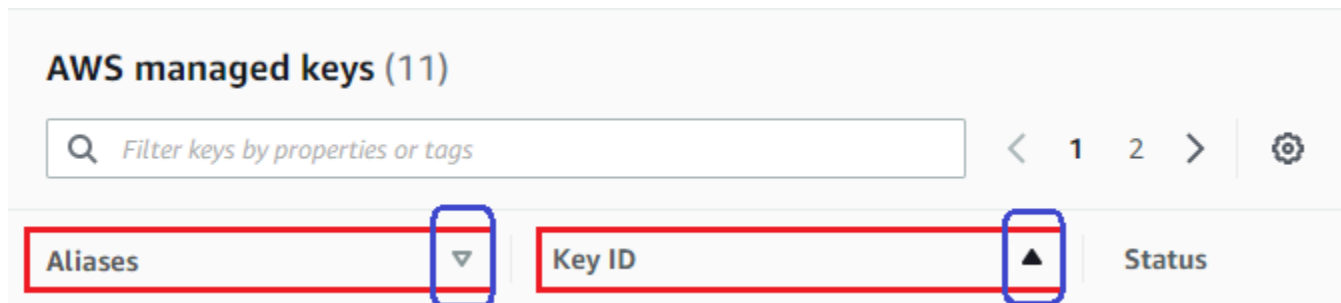
並べ替え

KMS キーを列の値で昇順または降順にソートできます。この機能は、テーブル内の KMS キーが現在のテーブルページに表示されていない場合も、すべての KMS キーをソートします。

ソート可能な列は、列名の横の矢印で示されます。AWS マネージドキー ページでは、[Aliases] (エイリアス) または [Key ID] (キー ID) でソートできます。[Customer managed keys] (カスタマーマネージドキー) ページでは、[Aliases] (エイリアス)、[Key ID] (キー ID)、または [Key type] (キータイプ) でソートできます。

昇順で並べ替えるには、矢印が上を向くように列見出しを選択します。降順で並べ替えるには、矢印が下を向くように列見出しを選択します。一度に 1 つの列のみでソートすることができます。

例えば、デフォルトのエイリアスではなく、キー ID により昇順で KMS キーをソートできます。



[Customer managed keys] (カスタマーマネージドキー) ページの KMS キーを [Key type] (キータイプ) で昇順にソートすると、すべての非対称キーがすべての対称キーの前に表示されます。

フィルター

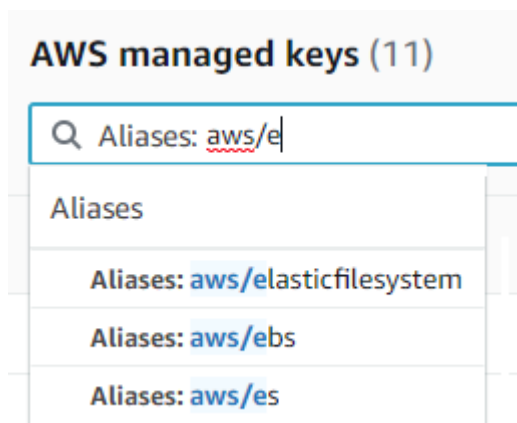
KMS キーは、プロパティ値またはタグでフィルタリングできます。フィルターは、現在のテーブルページに表示されていない場合でも、テーブル内のすべての KMS キーに適用されます。フィルターでは、大文字と小文字は区別されません。

フィルタリング可能なプロパティがフィルターボックスに一覧表示されます。AWS マネージドキー ページでは、エイリアスおよびキー ID でフィルタリングできます。[Customer managed keys] (カスタマーマネージドキー) ページでは、エイリアス、キー ID、キータイププロパティ、タグでフィルタリングできます。

- AWS マネージドキー ページでは、エイリアスおよびキー ID でフィルタリングできます。
- [Customer managed keys] (カスターマネージドキー) ページでは、タグ、エイリアス、キー ID、キータイプ、リージョナリティのプロパティでフィルタリングできます。

プロパティ値でフィルタリングするには、フィルター、プロパティ名の順に選択し、実際のプロパティ値のリストから選択します。タグでフィルタリングするには、タグキーを選択し、実際のタグ値のリストから選択します。プロパティまたはタグキーを選択した後、プロパティ値またはタグ値のすべてまたは一部を入力することもできます。選択を行う前に、結果のプレビューが表示されます。

例えば、aws/e を含むエイリアス名で KMS キーを表示するには、フィルターボックス、[Alias (エイリアス)]、タイプ aws/e の順に選択し、Enter または Return を押してフィルターを追加します。



推奨される KMS キーテーブルフィルター

非対称 KMS キーのフィルタリング

[Customer managed keys] (カスターマネージドキー) ページに非対称 KMS キーのみを表示するには、フィルターボックスをクリックして、[Key type] (キータイプ)、[Key type: Asymmetric] (キータイプ: 非対称) の順に選択します。非対称オプションは、テーブルに非対称 KMS キーがある場合にのみ表示されます。

マルチリージョンキーのフィルタリング

マルチリージョンキーのみを表示するには、[Customer managed keys] (カスターマネージドキー) ページで、フィルターボックス、[Regionality] (リージョナリティ)、[Regionality: Multi-Region] (リージョナリティ: マルチリージョン) の順に選択します。マルチリージョンキーオプションは、テーブルにマルチリージョンキーがある場合にのみ表示されます。

タグのフィルタリング

特定のタグを持つ KMS キーのみを表示するには、フィルターボックス、タグキーの順に選択し、実際のタグ値の中から選択します。タグ値のすべてまたは一部を入力することもできます。

結果のテーブルには、選択したタグを持つすべての KMS キーが表示されます。ただし、タグは表示されません。タグを表示するには、KMS キーのキー ID またはエイリアスを選択し、その詳細ページで [Tags] (タグ) タブを選択します。これらのタブは、一般設定セクションにあります。

このフィルターには、タグキーとタグ値の両方が必要です。タグキーのみを入力しても、その値のみを入力しても KMS キーは検索されません。タグキー、値のすべてまたは一部でタグをフィルタリングするには、[ListResourceTags](#) オペレーションを使用してタグ付けされた KMS キーを取得し、プログラミング言語のフィルタリング機能を使用します。

テキストでのフィルタリング

テキストを検索するには、フィルターボックスに、エイリアス、キー ID、キータイプ、タグキーのすべてまたは一部を入力します。(タグキーの選択後、タグ値を検索できます)。選択を行う前に、結果のプレビューが表示されます。

例えば、KMS キーをタグキーまたはフィルタリング可能なプロパティの `test` で表示するには、フィルターボックスの `test` を入力します。プレビューには、フィルターが選択する KMS キーが表示されます。この場合、`test` はエイリアスのプロパティにのみ表示されます。

KMS キーテーブルのカスタマイズ

AWS マネージドキー および のカスタマーマネージドキーページに表示されるテーブルは AWS Management Console、ニーズに合わせてカスタマイズできます。テーブル列、AWS KMS keys 各ページの の数 (ページサイズ)、テキストラップを選択できます。選択した設定は、確認すると保存され、ページを開くたびに再適用されます。

KMS キーテーブルをカスタマイズするには

1. AWS マネージドキー または [Customer managed keys] (カスタマーマネージドキー) ページで、ページの右上隅にある設定アイコン



を選択します。

2. [Preferences] (設定) ページで、必要な設定を選択してから [Confirm] (確認) を選択します。

ページサイズ設定を使用して、特に、スクロールしやすいデバイスを通常使用する場合、各ページに表示される KMS キーの数を増やすことを検討します。

表示されるデータ列は、テーブル、ジョブロール、アカウントとリージョンでの KMS キーのタイプによって異なる場合があります。次の表に、推奨される設定を示します。列の説明については、「[AWS KMS コンソールの使用](#)」を参照してください。

推奨される KMS キーテーブルの設定

KMS キーテーブルに表示される列をカスタマイズして、KMS キーに関する必要な情報を表示できます。

AWS マネージドキー

デフォルトでは、AWS マネージドキー テーブルにエイリアス、キー ID、ステータスの各列が表示されます。これらの列は、ほとんどのユースケースに最適です。

対称暗号化 KMS キー

AWS KMSが生成したキーマテリアルを持つ対称暗号化 KMS キーのみを使用する場合は、[Aliases] (エイリアス)、[Key ID] (キー ID)、[Status] (ステータス)、[Creation date] (作成日) の各列が最も役に立つと思われます。

非対称 KMS キー

非対称 KMS キーを使用する場合は、[Aliases] (エイリアス)、[Key ID] (キー ID)、[Status] (ステータス) 列に加えて、[Key type] (キータイプ)、[Key spec] (キー仕様)、[Key usage] (キーの使用法) 列を追加することを検討します。これらの列には、KMS キーが対称か非対称か、キーマテリアルのタイプ、KMS キーを暗号化または署名に使用できるかが表示されます。

HMAC KMS キー

HMAC KMS キーを使用する場合は、[Aliases] (エイリアス)、[Key ID] (キー ID)、[Status] (ステータス) 列に加えて、[Key type] (キーのタイプ)、[Key spec] (キーの仕様)、[Key usage] (キーの使用) 列を追加することを検討します。これらの列は、KMS キーが HMAC キーであるかどうかを示します。KMS キーをキー仕様またはキー使用状況でソートすることはできないため、エイリアスとタグを使用して HMAC キーを識別し、AWS KMS コンソールの[フィルター機能](#)を使用してエイリアスまたはタグでフィルタリングします。

インポートされたキーマテリアル

[インポートされたキーマテリアル](#)を持つ KMS キーの場合、[Origin] (オリジン) および[Expiration date] (有効期限日) 列を追加することを検討します。これらの列には、KMS キーのキーマテリアル

ルが によってインポートまたは生成されるかどうか、AWS KMS およびキーマテリアルの有効期限が切れるタイミングが表示されます。[Creation date] (作成日) フィールドには、KMS キーが (キーマテリアルなしで) 作成された日付が表示されます。キーマテリアルの特性を反映していません。

カスタムキーストアのキー

KMS キーが [カスタムキーストア](#) にある場合、[Origin] (オリジン) 列と [Custom key store ID] (カスタムキーストア ID) 列を追加することを検討します。これらの列は、KMS キーがカスタムキーストアにあることを示し、カスタムキーストアのタイプを表し、カスタムキーストアを識別します。

マルチリージョンキー

[マルチリージョンキー](#) がある場合、リージョナリティ列を追加することを検討します。これは、KMS キーが単一リージョンキー、[マルチリージョンのプライマリキー](#)、[マルチリージョンのレプリカキー](#) のいずれであるかを示します。

キーストアで KMS キーと AWS CloudHSM キーマテリアルを検索する

AWS CloudHSM キーストアを管理する場合は、各 AWS CloudHSM キーストアで KMS キーを識別する必要がある場合があります。例えば、次のタスクの一部を実行する必要がある場合があります。

- AWS CloudTrail ログでキーストアの KMS AWS CloudHSM キーを追跡します。
- キーストアを切断した場合の KMS AWS CloudHSM キーへの影響を予測します。
- キーストアを削除する前に、KMS AWS CloudHSM キーの削除をスケジュールします。

さらに、KMS キーのキーマテリアルとして機能する AWS CloudHSM クラスター内のキーを特定することもできます。は KMS キーとキーマテリアル AWS KMS を管理しますが、AWS CloudHSM クラスターの管理、HSMs とバックアップ、HSM 内の HSMs。キーマテリアルを監査したり、誤って削除されないようにしたり、KMS キーの削除後に HSM やクラスターバックアップから削除したりするために、キーを識別する必要がある場合があります。

キーストア内の KMS キーのすべての AWS CloudHSM キーマテリアルは、[kmsuserCrypto User](#) (CU) によって所有されます。は、 のみ表示できるキーラベル属性を AWS CloudHSM KMS キーの Amazon リソースネーム (ARN) AWS KMS に設定します。

KMS キーとキーマテリアルを検索するには、次のいずれかの方法を使用します。

- [キーストアで KMS AWS CloudHSM キーを検索する](#) — 1 つまたはすべてのキーストアで KMS AWS CloudHSM キーを識別する方法。
- [キーストアのすべての AWS CloudHSM キーを検索する](#) — AWS CloudHSM キーストアで KMS キーのキーマテリアルとして機能する、クラスター内すべてのキーを検索する方法。
- [KMS AWS CloudHSM キーのキーを検索する](#) — キーストア内の特定の KMS キーのキーマテリアルとして機能するクラスター内の AWS CloudHSM キーを見つける方法。
- [キーの KMS AWS CloudHSM キーを検索する](#) — クラスターで特定のキーの KMS キーを検索する方法。

キーストアで KMS AWS CloudHSM キーを検索する

AWS CloudHSM キーストアを管理する場合は、各 AWS CloudHSM キーストアで KMS キーを識別する必要がある場合があります。この情報を使用して、AWS CloudTrail ログで KMS キーオペレーションを追跡したり、KMS キーに対するカスタムキーストアの切断の影響を予測したり、AWS CloudHSM キーストアを削除する前に KMS キーの削除をスケジュールしたりできます。

キーストアで KMS AWS CloudHSM キーを検索するには (コンソール)

特定のキーストアで KMS AWS CloudHSM キーを検索するには、カスタマーマネージドキーページで、カスタムキーストア名またはカスタムキーストア ID フィールドの値を表示します。任意の AWS CloudHSM キーストアで KMS キーを識別するには、オリジン値が の KMS キーを探します AWS CloudHSM。オプションの列をディスプレイに追加するには、ページの右上隅にある歯車アイコンを選択します。

キーストアで KMS AWS CloudHSM キーを検索するには (API)

キーストアで KMS AWS CloudHSM キーを検索するには、[ListKeys](#) オペレーションと [DescribeKey](#) オペレーションを使用して、CustomKeyStoreId値でフィルタリングします。以下の各例を実行する前に、架空のカスタムキーストア ID の値を有効な値に置き換えます。

Bash

特定のキーストアで KMS AWS CloudHSM キーを検索するには、アカウントとリージョン内のすべての KMS キーを取得します。次に、カスタムキーストア ID でフィルタをかけます。

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;  
do aws kms describe-key --key-id $key |
```

```
grep '"CustomKeyStoreId": "cks-1234567890abcdef0"' --context 100; done
```

アカウントとリージョンの任意のキーストアで KMS AWS CloudHSM キーを取得するには、の値 CustomKeyStoreType でを検索します AWS_CloudHSM。

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;  
do aws kms describe-key --key-id $key |  
grep '"CustomKeyStoreType": "AWS_CloudHSM"' --context 100; done
```

PowerShell

特定の AWS CloudHSM キーストアで KMS キーを検索するには、[Get-KmsKeyList](#) コマンドレットと [Get-KmsKey](#) コマンドレットを使用して、アカウントとリージョン内のすべての KMS キーを取得します。次に、カスタムキーストア ID でフィルタをかけます。

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreId -eq  
'cks-1234567890abcdef0'
```

アカウントとリージョンの任意のキーストアで KMS AWS CloudHSM キーを取得するには、CustomKeyStoreType の値をフィルタリングします AWS_CLOUDHSM。

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyStoreType -eq 'AWS_CLOUDHSM'
```

キーストアのすべての AWS CloudHSM キーを検索する

キーストアのキーマテリアル AWS CloudHSM として機能する AWS CloudHSM クラスター内のキーを識別できます。これを行うには、CloudHSM CLI 内で [key list](#) コマンドを使用します。

キーリストコマンドを使用して、キー AWS KMS の AWS CloudHSM を検索することもできます。が AWS CloudHSM クラスターで KMS キーのキーマテリアル AWS KMS を作成すると、キーラベルに KMS キーの Amazon リソースネーム (ARN) が書き込まれます。[key list] コマンドは、key-reference と label を返します。

メモ

次の手順では、AWS CloudHSM クライアント SDK 5 コマンドラインツール [CloudHSM CLI](#) を使用します。CloudHSM CLI は、key-handle を key-reference に置き換えます。

2025 年 1 月 1 日、AWS CloudHSM はクライアント SDK 3 コマンドラインツール、CloudHSM 管理ユーティリティ (CMU)、およびキー管理ユーティリティ (KMU) のサポートを終了します。Client SDK 3 コマンドラインツールと Client SDK 5 コマンドラインツールの違いの詳細については、「AWS CloudHSM ユーザーガイド」の「[Client SDK 3 CMU および KMU から Client SDK 5 CloudHSM CLI への移行](#)」を参照してください。

この手順を実行するには、CU kmsuser としてログインできるように、AWS CloudHSM キーストアを一時的に切断する必要があります。

1. AWS CloudHSM キーストアがまだ切断されていない場合は切断し、「」の説明に従って kmsuser としてログインします [切断してログインする方法](#)。

Note

カスタムキーストアが切断されている間は、カスタムキーストアで KMS キーを作成したり、暗号化オペレーションで既存の KMS キーを使用したりする試みはすべて失敗します。このオペレーションにより、ユーザーが機密データを保存したりアクセスしたりすることを防ぐことができます。

2. CloudHSM CLI の [key list](#) コマンドを使用して、AWS CloudHSM クラスタに存在する現在のユーザーのすべてのキーを検索します。

デフォルトでは、現在ログインしているユーザーのキーは 10 個だけ表示され、出力には key-reference と label のみ表示されます。その他のオプションについては、「AWS CloudHSM ユーザーガイド」の「[key list](#)」を参照してください。

```
aws-cloudhsm > key list
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000000123",
        "attributes": {
          "label": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
      },
      {
```



```
    "key-reference": "0x00000000000000456",
    "attributes": {
      "label": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
    }
  },
  ...8 keys later...
],
"total_key_count": 56,
"returned_key_count": 10,
"next_token": "10"
}
```

3. 「」の説明に従って、ログアウトし、AWS CloudHSM キーストアを再接続します [ログアウトして再接続する方法](#)。

キーの KMS AWS CloudHSM キーを検索する

がクラスター内でkmsuser所有しているキーのキーリファレンスまたは ID がわかっている場合は、その値を使用して、キーストア内の関連する KMS AWS CloudHSM キーを識別できます。

が AWS CloudHSM クラスターで KMS キーのキーマテリアル AWS KMS を作成すると、キーラベルに KMS キーの Amazon リソースネーム (ARN) が書き込まれます。ラベル値を変更しない限り、CloudHSM CLI の [key list](#) コマンドを使用して、AWS CloudHSM キーに関連付けられた KMS キーを特定できます。

メモ

次の手順では、AWS CloudHSM クライアント SDK 5 コマンドラインツール [CloudHSM CLI](#) を使用します。CloudHSM CLI は、key-handle を key-reference に置き換えます。2025 年 1 月 1 日、AWS CloudHSM はクライアント SDK 3 コマンドラインツール、CloudHSM 管理ユーティリティ (CMU)、およびキー管理ユーティリティ (KMU) のサポートを終了します。Client SDK 3 コマンドラインツールと Client SDK 5 コマンドラインツールの違いの詳細については、「AWS CloudHSM ユーザーガイド」の「[Client SDK 3 CMU および KMU から Client SDK 5 CloudHSM CLI への移行](#)」を参照してください。

これらの手順を実行するには、CU kmsuser としてログインできるように、AWS CloudHSM キーストアを一時的に切断する必要があります。

Note

カスタムキーストアが切断されている間は、カスタムキーストアで KMS キーを作成したり、暗号化オペレーションで既存の KMS キーを使用したりする試みはすべて失敗します。このオペレーションにより、ユーザーが機密データを保存したりアクセスしたりすることを防ぐことができます。

トピック

- [キーリファレンスに関連付けられた KMS キーを特定する](#)
- [バックアップキー ID に関連付けられた KMS キーを特定する](#)

キーリファレンスに関連付けられた KMS キーを特定する

次の手順は、CloudHSM CLI の [\[key list\]](#) コマンドを key-reference 属性フィルターと併せて使用することにより、AWS CloudHSM キーストア内の特定の KMS キーのキーマテリアルとして機能するクラスター内のキーを検索する方法を示します。

1. AWS CloudHSM キーストアがまだ切断されていない場合は切断し、「」の説明に従って kmsuser としてログインします [切断してログインする方法](#)。
2. CloudHSM CLI の [\[key list\]](#) コマンドを使用して、key-reference 属性でフィルタリングします。一致するキーのすべての属性とキー情報を含める verbose 引数を指定します。verbose 引数を指定しない場合、[key list] オペレーションは一致するキーのキーリファレンスとラベル属性のみを返します。

このコマンドを実行する前に、サンプル key-reference をお使いのアカウントにある有効な値に置き換えてください。

```
aws-cloudhsm > key list --filter attr.key-reference="0x0000000000120034" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000120034",
        "key-info": {
          "key-owners": [
            {
```

```

        "username": "kmsuser",
        "key-coverage": "full"
    },
    ],
    "shared-users": [],
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "aes",
    "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "id": "0xbacking-key-id",
    "check-value": "0x29bbd1",
    "class": "my_test_key",
    "encrypt": true,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": false,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": false,
    "trusted": false,
    "unwrap": true,
    "verify": false,
    "wrap": true,
    "wrap-with-trusted": false,
    "key-length-bytes": 32
}
}
],
"total_key_count": 1,
"returned_key_count": 1
}
}

```

3. 「」の説明に従って、ログアウトし、AWS CloudHSM キーストアを再接続します [ログアウトして再接続する方法](#)。

バックアップキー ID に関連付けられた KMS キーを特定する

AWS CloudHSM キーストアの KMS キーを使用した暗号化オペレーションのすべての CloudTrail ログエントリには、`customKeyStoreId` および `backingKeyId` を含む `additionalEventData` フィールドが含まれます。`backingKeyId` フィールドで返される値は、CloudHSM キー `id` 属性と相互に関連付けられています。`id` 属性で [\[key list\]](#) オペレーションをフィルタリングすることで、特定の `backingKeyId` に関連付けられた KMS キーを特定できます。

1. AWS CloudHSM キーストアがまだ切断されていない場合は切断し、「」の説明に従って `kmsuser` としてログインします [切断してログインする方法](#)。
2. CloudHSM CLI の [\[key list\]](#) コマンドを属性フィルターと併せて使用することで、AWS CloudHSM キーストア内の特定の KMS キーのキーマテリアルとして機能するクラスター内のキーを検索します。

次の例は、`id` 属性でフィルタリングする方法を示します。AWS CloudHSM は `id` 値を 16 進値として認識します。`id` 属性でキーリストオペレーションをフィルタリングするには、まず CloudTrail ログエントリで識別した `backingKeyId` 値を AWS CloudHSM 認識する形式に変換する必要があります。

- a. 次の Linux コマンドを使用して、`backingKeyId` を 16 進数表現に変換します。

```
echo backingKeyId | tr -d '\n' | xxd -p
```

次の例は、`backingKeyId` バイト配列を 16 進数表現に変換する方法を示します。

```
echo 5890723622dc15f699aa9ab2387a9f744b2b884c18b2186ee8ada4f556a2eb9d | tr -d '\n' | xxd -p  
353839303732333632326463313566363939616139616232333837613966373434623262383834633138623
```

- b. `backingKeyId` の 16 進数表現の先頭に `0x` を付加します。

```
0x353839303732333632326463313566363939616139616232333837613966373434623262383834633138623
```

- c. 変換された `backingKeyId` 値を使用して、`id` 属性でフィルタリングします。一致するキーのすべての属性とキー情報を含める `verbose` 引数を指定します。`verbose` 引数を指定しない場合、`[key list]` オペレーションは一致するキーのキーリファレンスとラベル属性のみを返します。

```
aws-cloudhsm > key list --filter
attr.id="0x353839303732333632326463313566363939616139616232333837613966373434623262383
--verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
          "id":
"0x353839303732333632326463313566363939616139616232333837613966373434623262383463313
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": false,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": true,
          "verify": false,
```

```
        "wrap": true,  
        "wrap-with-trusted": false,  
        "key-length-bytes": 32  
    }  
}  
],  
"total_key_count": 1,  
"returned_key_count": 1  
}  
}
```

3. 「」の説明に従って、ログアウトし、AWS CloudHSM キーストアを再接続します[ログアウトして再接続する方法](#)。

KMS AWS CloudHSM キーのキーを検索する

キーストアの KMS キーの KMS AWS CloudHSM キー ID を使用して、キーマテリアルとして機能する AWS CloudHSM クラスター内のキーを識別できます。

が AWS CloudHSM クラスターで KMS キーのキーマテリアル AWS KMS を作成すると、キーラベルに KMS キーの Amazon リソースネーム (ARN) が書き込まれます。ラベル値を変更しない限り、CloudHSM CLI の [\[key list\]](#) コマンドを使用して、KMS キーのキーマテリアルのキーリソースおよび ID を取得できます。

AWS CloudHSM キーストアの KMS キーを使用した暗号化オペレーションのすべての CloudTrail ログエントリには、customKeyIdおよびを含む additionalEventDataフィールドが含まれますbackingKeyId。backingKeyId フィールドで返される値は、id AWS CloudHSM キー属性です。KMS キー ARN でキーリスト AWS CloudHSM CLI オペレーションをフィルタリングして、特定の KMS キーに関連付けられた CloudHSM キーid属性を識別できます。

この手順を実行するには、CU kmsuser としてログインできるように、AWS CloudHSM キーストアを一時的に切断する必要があります。

メモ

次の手順では、AWS CloudHSM クライアント SDK 5 コマンドラインツール [CloudHSM CLI](#) を使用します。CloudHSM CLI は、key-handle を key-reference に置き換えます。2025 年 1 月 1 日、AWS CloudHSM はクライアント SDK 3 コマンドラインツール、CloudHSM 管理ユーティリティ (CMU)、およびキー管理ユーティリティ (KMU) のサポートを終了します。Client SDK 3 コマンドラインツールと Client SDK 5 コマンドライン

ツールの違いの詳細については、「AWS CloudHSM ユーザーガイド」の「[Client SDK 3 CMU および KMU から Client SDK 5 CloudHSM CLI への移行](#)」を参照してください。

1. AWS CloudHSM キーストアがまだ切断されていない場合は切断し、「」の説明に従って kmsuser としてログインします [切断してログインする方法](#)。

 Note

カスタムキーストアが切断されている間は、カスタムキーストアで KMS キーを作成したり、暗号化オペレーションで既存の KMS キーを使用したりする試みはすべて失敗します。このオペレーションにより、ユーザーが機密データを保存したりアクセスしたりすることを防ぐことができます。

2. CloudHSM CLI の [key list](#) コマンドを使用し、でフィルタリング `label` して、AWS CloudHSM クラスター内の特定のキーの KMS キーを検索します。一致するキーのすべての属性とキー情報を含める `verbose` 引数を指定します。verbose 引数を指定しない場合、[key list] オペレーションは、一致するキーのキーリファレンスとラベル属性のみを返します。

次の例は、KMS キー ARN を格納する `label` 属性を用いたフィルタリング方法を示します。このコマンドを実行する前に、KMS キー ARN のサンプル値をお使いのアカウントにある有効な値に置き換えてください。

```
aws-cloudhsm > key list --filter attr.label="arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000120034",
        "key-info": {
          "key-owners": [
            {
              "username": "kmsuser",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        }
      }
    ]
  }
}
```

```
    },
    "attributes": {
      "key-type": "aes",
      "label": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "id": "0xbacking-key-id",
      "check-value": "0x29bbd1",
      "class": "my_test_key",
      "encrypt": true,
      "decrypt": true,
      "token": true,
      "always-sensitive": true,
      "derive": false,
      "destroyable": true,
      "extractable": false,
      "local": true,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": true,
      "sign": false,
      "trusted": false,
      "unwrap": true,
      "verify": false,
      "wrap": true,
      "wrap-with-trusted": false,
      "key-length-bytes": 32
    }
  ],
  "total_key_count": 1,
  "returned_key_count": 1
}
```

3. 「」の説明に従って、ログアウトし、AWS CloudHSM キーストアを再接続します[ログアウトして再接続する方法](#)。

キーの有効化と無効化

カスタマーマネージドキーの有効化と無効化を行うことができます。KMS キー を作成すると、そのキーはデフォルトで有効になります。KMS キーを無効にすると、再度有効にするまで[暗号化オペレーション](#)で使用できなくなります。

一時的で簡単に元に戻せるため、KMS キーを無効にすることは、破壊的で元に戻せないアクションである KMS キーを削除することの安全な代替手段です。KMS キーを削除することを検討している場合は、まずそれを無効にして、[CloudWatch アラーム](#)または、暗号化されたデータを復号するためにキーを使用する必要がないことを確実にするための同様のメカニズムを設定します。

KMS キーを無効にするとそのキーはただちに使用できなくなります (結果整合性の影響を受ける)。ただし、KMS キーで保護された[データキー](#)を使って暗号化されたリソースは、KMS キーがデータキーの復号化などに再び使用されるまでは、影響を受けません。この問題は AWS のサービスに影響します。その多くは、データキーを使用してリソースを保護します。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

を有効[AWS マネージドキー](#)または無効にすることはできません[AWS 所有のキー](#)。AWS マネージドキー は、[を使用するサービス AWS KMS](#)で使用するために完全に有効になっています。AWS 所有のキー は、それらを所有するサービスによってのみ管理されます。

Note

AWS KMS は、カスタマーマネージドキーのキーマテリアルを無効にしている間はローテーションしません。詳細については、「[キーローテーションの仕組み](#)」を参照してください。

AWS KMS コンソールの使用

AWS KMS コンソールを使用して、[カスタマーマネージドキー](#)を有効または無効にできます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. 有効または無効にする KMS キーのチェックボックスをオンにします。

5. KMS キーを有効にするには、[Key actions (キーアクション)]、[Enable (有効)] の順に選択します。KMS キーを無効にするには、[Key actions (キーアクション)]、[Disable (無効)] の順に選択します。

AWS KMS API の使用

[EnableKey](#) オペレーションは、無効になっている を有効にします AWS KMS key。以下の例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。key-id パラメータは必須です。

このオペレーションはどのような出力も返しません。キーのステータスを確認するには、[DescribeKey](#) オペレーションを使用します。

```
$ aws kms enable-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

[DisableKey](#) オペレーションは有効な KMS キーを無効にします。key-id パラメータは必須です。

```
$ aws kms disable-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

このオペレーションはどのような出力も返しません。キーのステータスを確認するには、[DescribeKey](#) オペレーションを使用し、Enabled フィールドを参照してください。

```
$ aws kms describe-key --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "KeyMetadata": {
    "Origin": "AWS_KMS",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Description": "",
    "KeyManager": "CUSTOMER",
    "MultiRegion": false,
    "Enabled": false,
    "KeyState": "Disabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "CreationDate": 1502910355.475,
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
```

```
    "SYMMETRIC_DEFAULT"  
  ]  
}  
}
```

ローテーション AWS KMS keys

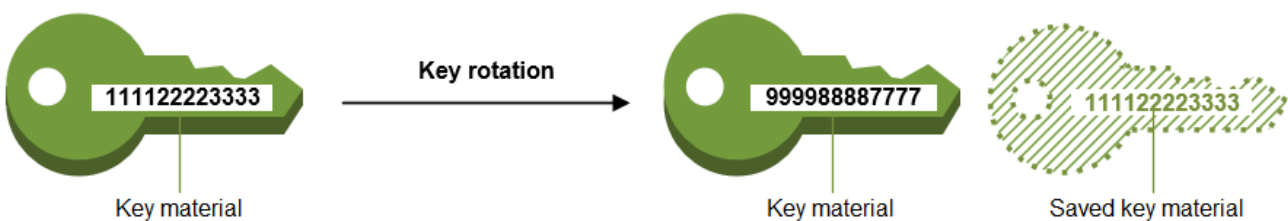
[カスタマーマネージドキー](#)の新しい暗号化マテリアルを作成するには、新しい KMS キーを作成し、アプリケーションまたはエイリアスを変更して新しい KMS キーを使用します。または、自動キーローテーションを有効にするか、オンデマンドローテーションを実行することで、既存の KMS キーに関連付けられたキーマテリアルをローテーションすることもできます。

デフォルトでは、KMS キーの自動キーローテーションを有効にすると、は KMS キーの新しい暗号化マテリアルを毎年 AWS KMS 生成します。カスタムを指定[rotation-period](#)して、キーマテリアルを AWS KMS ローテーションする自動キーローテーションを有効にしてからの日数と、それ以降の各自動ローテーション間の日数を定義することもできます。キーマテリアルローテーションをすぐに開始する必要がある場合は、自動キーローテーションが有効化されているかどうかにかかわらず、オンデマンドローテーションを実行できます。オンデマンドローテーションは、既存の自動ローテーションスケジュールには影響を与えません。

Amazon CloudWatch AWS CloudTrailおよび AWS Key Management Service コンソールで KMS キーのキーマテリアルの[ローテーションを追跡](#)できます。[GetKeyRotationStatus](#) オペレーションを使用して、特定の KMS キーの自動ローテーションが有効になっているかどうかを確認したり、進行中のオンデマンドローテーションの状況をチェックすることができます。[ListKeyRotations](#) オペレーションを使用して、実施済みのローテーションの詳細を確認できます。

キーローテーションは、暗号化オペレーションで使用される暗号化シークレットである現在のキーマテリアルのみを変更します。ローテーションされた KMS キーを使用して暗号化テキストを復号すると、は暗号化に使用されたキーマテリアル AWS KMS を使用します。復号オペレーションで特定のキーマテリアルを選択することはできません。は正しいキーマテリアル AWS KMS を自動的に選択します。は適切なキーマテリアルで AWS KMS 透過的に復号するため、ローテーションされた KMS キーは、コードを変更 AWS のサービス せずにアプリケーションで安全に使用できます。

KMS キーは、そのキーマテリアルの変更の有無や回数に関わらず、同じ論理リソースのままです。次のイメージに示されているように、KMS キーのプロパティは変更されません。



Key ID = 1234abcd-12ab-34cd-56ef-1234567890ab

Key ID = 1234abcd-12ab-34cd-56ef-1234567890ab

新しい KMS キーを作成し、元の KMS キーの代わりに使用することを決定することができます。これには、既存の KMS キーでキーマテリアルをローテーションするのと同じ効果があり、多くの場合、[手動キーローテーション](#)を使用します。手動ローテーションは、[非対称 KMS キー](#)、[HMAC KMS キー](#)、[カスタムキーストアの KMS キー](#)、インポートされたキーマテリアルを持つマルチリージョン KMS キーなど、自動キーローテーションまたはオンデマンドキーローテーションの対象ではない KMS キーをローテーションする場合に適しています。 [??? ???](#)

Note

キーローテーションは、KMS キーが保護するデータには影響しません。KMS キーが生成したデータキーをローテーションしたり、KMS キーによって保護されたデータを再暗号化したりすることはありません。キーローテーションは、侵害されたデータキーの影響を軽減しません。

キーローテーションと料金

AWS KMS は、KMS キー用に維持されるキーマテリアルの初回および 2 回目のローテーションに対して月額料金を請求します。この追加課金は 2 回目のローテーションで上限となり、それ以降のローテーションには課金されません。詳細については、「[AWS Key Management Service の料金表](#)」を参照してください。

Note

[AWS Cost Explorer Service](#) を使用して、キーストレージ料金の内訳を表示できます。例えば、[使用タイプ] に \$REGION-KMS-Keys を指定し、データを [API オペレーション] でグループ化することによって、ビューをフィルターし、現在のローテーションされた KMS キーとして課金されたキーの合計料金を表示できます。

過去の期間のレガシー Unknown API オペレーションのインスタンスが引き続き表示される場合があります。

キーローテーションとクォータ

各 KMS キーは、キーリソースのクォータを計算するときに、ローテーションされたキーマテリアルのバージョン数に関係なく、1 つのキーとしてカウントされます。

キーマテリアルとローテーションの詳細については、[AWS Key Management Service 暗号化の詳細](#)を参照してください。

トピック

- [KMS キーをローテーションする理由](#)
- [キーローテーションの仕組み](#)
- [自動キーローテーションを有効にする](#)
- [自動キーローテーションを無効にする](#)
- [オンデマンドキーローテーションの実行](#)
- [ローテーションとキーマテリアルを一覧表示する](#)
- [キーを手動でローテーションする](#)
- [マルチリージョンキーセットのプライマリキーを変更する](#)

KMS キーをローテーションする理由

暗号化のベストプラクティスでは、が AWS KMS 生成するデータキーなど、データを直接暗号化する [キー](#) の広範な再利用は推奨されません。256 ビットデータキーが数百万のメッセージを暗号化すると、キーが枯渇し、微小なパターンを含む暗号文を生成し始める可能性があり、ずる賢い人物が悪用して、キーの中のビットが発見される可能性があります。このキーの枯渇を軽減するために、データキーを 1 回または数回使用することをお勧めします。

ただし、KMS キーは、ラップキー (キー暗号化キーとも呼ばれます) として最もよく使用されます。データを暗号化する代わりに、ラップキーはデータを暗号化するデータキーを暗号化します。そのため、これらはデータキーよりもはるかに使用頻度が低く、キーが枯渇するほど多数回再利用されることはほとんどありません。

このように枯渇するリスクは極めて低いものの、ビジネスルールや契約、政府の規制により、KMS キーのローテーションが必要になる場合があります。KMS キーのローテーションを強制する場合は、サポートされている場合は自動キーローテーションを使用し、自動ローテーションがサポートされていない場合はオンデマンドローテーションを使用し、自動キーローテーションもオンデマンドキーローテーションもサポートされていない場合は手動キーローテーションを使用することをお勧めします。

キーマテリアルローテーション機能をテストするため、またはオートメーションスクリプトを検証するために、オンデマンドローテーションの実行を検討するのもよいでしょう。計画外のローテーションにはオンデマンドローテーションを使用し、可能な限りカスタムローテーション [期間で自動キーローテーション](#) を使用することをお勧めします。

キーローテーションの仕組み

AWS KMS キーローテーションは、透過的で使いやすいように設計されています。は、[カスタマー マネージドキーに対してのみ、オプションの自動キーローテーションとオンデマンドキーローテーション](#) AWS KMS をサポートします。

自動キーローテーション

AWS KMS は、ローテーション期間で定義された次のローテーション日に KMS キーを自動的にローテーションします。更新を覚えている、またはスケジュールする必要はありません。

自動キーローテーションは、(AWS_KMS オリジン) AWS KMS を生成するキーマテリアルを持つ対称暗号化 KMS キーでのみサポートされます。

カスタマー マネージド KMS キーでは、自動ローテーションはオプションです。AWS KMS は常に AWS、マネージド KMS キーのキーマテリアルを毎年ローテーションします。AWS 所有の KMS キーのローテーションは、キーを所有 AWS のサービス する によって管理されます。

オンデマンドローテーション

自動キーローテーションが有効化されているかどうかにかかわらず、KMS キーに関連付けられたキーマテリアルのローテーションを直ちに開始します。

オンデマンドキーローテーションは、(AWS_KMS オリジン) AWS KMS を生成するキーマテリアルを持つ対称暗号化 KMS キーと、インポートされたキーマテリアル (EXTERNAL オリジン) を持つ単一リージョンの対称暗号化 KMS キーでサポートされています。

手動ローテーション

次のタイプの KMS キーでは、自動キーローテーションもオンデマンドキーローテーションもサポートされていませんが、[これらの KMS キーは手動でローテーション](#)できます。

- [非対称 KMS キー](#)
- [HMAC KMS キー](#)
- [カスタムキーストアの KMS キー](#)
- [インポートされたキーマテリアルを持つマルチリージョン KMS キー](#)

キーマテリアルの管理

AWS KMS は、キーローテーションが無効になっている場合でも、AWS_KMSオリジンを持つ KMS キーのすべてのキーマテリアルを保持します。は、KMS キーを削除した場合にのみキーマテリアル AWS KMS を削除します。

オEXTERNALリジンを持つ対称暗号化キーのキーマテリアルを管理します。 [DeleteImportedKeyMaterial](#) オペレーションを使用して任意のキーマテリアルを削除するか、マテリアルをインポートするときに有効期限を設定できます。KMS キーは、マテリアルの有効期限が切れるか削除されるとすぐに使用できなくなります。

キーマテリアルの使用

ローテーションされた KMS キーを使用してデータを暗号化する場合、は現在のキーマテリアル AWS KMS を使用します。暗号文の復号にローテーションされた KMS キーを使用する場合、AWS KMS は暗号化に使用したものと同一バージョンのキーマテリアルを使用します。復号オペレーションに特定のバージョンのキーマテリアルを選択することはできません。は正しいバージョン AWS KMS を自動的に選択します。

ローテーション期間

ローテーション期間は、キーマテリアルを AWS KMS ローターションする自動キーローテーションを有効にしてからの日数と、それ以降の各自動キーローテーション間の日数を定義します。自動キーローテーションを有効化したときに `RotationPeriodInDays` の値を指定しない場合、デフォルト値は 365 日となります。

[kms:RotationPeriodInDays](#) 条件キーを使用して、プリンシパルが `RotationPeriodInDays` パラメータに指定できる値を制限することができます。

ローテーション日

ローテーション日は、自動 (スケジュールされた) ローターションまたはオンデマンドキーローテーションの結果として KMS キーの現在のキーマテリアルが更新された日付を反映します。

ローテーション日

AWS KMS は、ローテーション期間で定義されたローテーション日に KMS キーを自動的にローテーションします。デフォルトのローテーション期間は 365 日です。

カスターマネージドキー

自動キーローテーションは、[カスターマネージドキー](#)ではオプションであり、いつでも有効化および無効化できるため、ローテーション日は、ローテーションの最終有効化日によって異なります。自動キーローテーションを有効化した後にキーのローテーション期間を変更すると、ローテーション日が変わる可能性があります。ローテーション日は、キーの有効期間にわたって何度も変更できます。

たとえば、2022 年 1 月 1 日にカスターマネージドキーを作成した後、2022 年 3 月 15 日にローテーション期間をデフォルトの 365 日に設定して自動キーローテーションを有効化する

と、AWS KMS は、まず 2023 年 3 月 15 日、その次は 2024 年 3 月 15 日、以降 365 日ごとにキーマテリアルをローテーションします。

以下の各例ではいずれも、キーの自動ローテーションが有効で、ローテーション期間がデフォルトの 365 日に設定されていることを前提にしています。これらの例は、キーのローテーション期間に影響を与える可能性のある特別なケースを示しています。

- キーローテーションの無効化 — 任意の時点で[自動キーローテーションを無効](#)にすると、KMS キーは、ローテーションが無効になったときに使用していたバージョンのキーマテリアルを使用し続けます。自動キーローテーションを再度有効にすると、は新しいローテーションが有効な日付に基づいてキーマテリアルを AWS KMS ローテーションします。
- 無効 KMS キー — KMS キーが無効になっている間は、ローテーション AWS KMS しません。ただし、キーローテーションのステータスは変更されず、KMS キーが無効の間は変更することができません。KMS キーを再度有効にすると、キーマテリアルが最後にスケジュールされたローテーション日を過ぎていている場合、はそれをすぐに AWS KMS ローテーションします。キーマテリアルが最後にスケジュールされたローテーション日を過ぎていない場合、は元のキーローテーションスケジュール AWS KMS を再開します。
- 削除保留中の KMS キー — 削除保留中の KMS キーはローテーション AWS KMS されません。キーローテーションのステータスは false に設定されています。削除が保留中の場合は変更することができません。削除をキャンセルすると、以前のキーローテーションのステータスが元に戻ります。キーマテリアルが最後にスケジュールされたローテーション日を過ぎていている場合、はそれをすぐに AWS KMS ローテーションします。キーマテリアルが最後にスケジュールされたローテーション日を過ぎていない場合、は元のキーローテーションスケジュール AWS KMS を再開します。

AWS マネージドキー

AWS KMS は AWS マネージドキー 毎年 (約 365 日) 自動的にローテーションします。[AWS マネージドキー](#)のキーローテーションを有効化または無効化することはできません。

のキーマテリアル AWS マネージドキー は、最初に作成日から 1 年後にローテーションされ、その後は毎年 (最後のローテーションから約 365 日) ローテーションされます。

Note

2022 年 5 月、のローテーションスケジュールを 3 年ごと (約 1,095 日) AWS マネージドキー から 1 年ごと (約 365 日) AWS KMS に変更しました。

AWS 所有のキー

AWS 所有のキーのキーローテーションを有効化または無効化することはできません。の[キーローテーション](#)戦略 AWS 所有のキー は、 がキーを作成および管理する AWS サービスによって決まります。詳細については、サービスのユーザーガイドまたはデベロッパーガイドの「保管時の暗号化」トピックを参照してください。

マルチリージョンキーのローテーション

自動ローテーションを有効または無効にし、オAWS_KMSリジンを持つ対称暗号化[マルチリージョンキー](#)でキーマテリアルのオンデマンドローテーションを実行できます。キーローテーションは、マルチリージョンキーの[共有プロパティ](#)です。

自動キーローテーションはプライマリキーでのみ、有効または無効にできます。オンデマンドローテーションは、プライマリキーでのみ開始できます。

- がマルチリージョンキーを AWS KMS 同期すると、キーローテーションプロパティ設定がプライマリキーから関連するすべてのレプリカキーにコピーされます。
- はキーマテリアルを AWS KMS ローテーションすると、プライマリキーの新しいキーマテリアルを作成し、リージョンの境界を越えて関連するすべてのレプリカキーに新しいキーマテリアルをコピーします。キーマテリアルが暗号化 AWS KMS されていないままになることはありません。このステップは、暗号化オペレーションでキーが使用される前にキーマテリアルが完全に同期されるよう、慎重に制御されます。
- AWS KMS は、そのキーマテリアルがプライマリキーとそのすべてのレプリカキーで使用可能になるまで、新しいキーマテリアルを使用してデータを暗号化しません。
- ローテーションされたプライマリキーをレプリケートすると、新しいレプリカキーに、現在のキーマテリアルと関連するマルチリージョンキーのキーマテリアルの以前のすべてのバージョンが含まれます。

このパターンにより、関連するマルチリージョンキーが完全に相互運用可能であることが保証されます。すべてのマルチリージョンキーは、キーが作成される前に暗号化テキストが暗号化されていても、関連するマルチリージョンキーによって暗号化された暗号化テキストを復号できます。

AWS サービス

AWS サービスのサーバー側の暗号化に使用する[カスタマーマネージドキー](#)で、自動キーローテーションを有効にできます。年間ローテーションは透過的で、AWS のサービスと互換性があります。

キーローテーションのモニタリング

が [AWS マネージドキー](#) または [カスターマネージドキー](#) のキーマテリアルを AWS KMS ローターションすると、Amazon EventBridge に KMS CMK Rotation イベントを書き込み、[RotateKey イベント](#) をログに書き込みます AWS CloudTrail。これらのレコードを使用して、KMS キーがローテーションされたことを確認できます。

AWS Key Management Service コンソールを使用して、KMS キーの残りのオンデマンドローテーションの数と、完了したすべてのキーマテリアルローテーションのリストを表示できます。

[ListKeyRotations](#) オペレーションを使用して、実施済みのローテーションの詳細を確認できます。

結果整合性

キーローテーションは、他の AWS KMS 管理オペレーションと同様に、結果整合性の影響下にあります。新しいキーマテリアルが AWS KMS 全体で使用可能になるまで、若干の遅延が生じることがあります。ただし、キーマテリアルのローテーションにより、暗号化オペレーションが中断または遅延することはありません。新しいキーマテリアルが AWS KMS 全体で使用可能になるまで、現在のキーマテリアルが暗号化オペレーションで使用されます。マルチリージョンキーのキーマテリアルが自動的にローテーションされると、は、関連するマルチリージョンキーを持つすべてのリージョンで新しいキーマテリアルが利用可能になるまで、現在のキーマテリアル AWS KMS を使用します。

自動キーローテーションを有効にする

デフォルトでは、KMS キーの自動キーローテーションを有効にすると、は KMS キーの新しい暗号化マテリアルを毎年 AWS KMS 生成します。カスタムを指定 [rotation-period](#) して、キーマテリアルを AWS KMS ローターションする自動キーローテーションを有効にしてからの日数と、それ以降の各自動ローテーション間の日数を定義することもできます。

自動キーローテーションには次の利点があります。

- [キー ID](#)、[キー ARN](#)、リージョン、ポリシー、アクセス許可などの KMS キーのプロパティは、キーがローテーションされても変更されません。
- KMS キーのキー ID またはキー ARN を参照するアプリケーションまたはエイリアスを変更する必要はありません。
- キーマテリアルのローテーションは、どの AWS のサービスでの KMS キーの使用にも影響しません。

- キーローテーションを有効にすると、は AWS KMS ローターション期間で定義された次のローテーション日に KMS キーを自動的にローテーションします。更新を覚えている、またはスケジュールする必要はありません。

自動キーローテーションは、AWS KMS コンソールで有効にすることも、[EnableKeyRotation](#) オペレーションを使用して有効にすることもできます。自動キーローテーションを有効にするには、`kms:EnableKeyRotation` アクセス許可が必要です。アクセス AWS KMS 許可の詳細については、「」を参照してください[アクセス許可に関するリファレンス](#)。

AWS KMS コンソールの使用

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[Customer managed keys] (カスターマネージドキー) を選択します。(AWS マネージドキーのローテーションを有効化または無効化することはできません。これらのローテーションは毎年自動的に行われます。)
4. KMS キーのエイリアスまたは キー ID を選択します。
5. [キーローテーション] タブを選択します。

[Key rotation] (キーローテーション) タブは、[マルチリージョン](#)の対称暗号化 KMS キーなど、AWS KMS が生成したキーマテリアルを持つ対称暗号化 KMS キー ([Origin] (オリジン) が [AWS_KMS]) の詳細ページにのみ表示されます。

非対称 KMS キー、HMAC KMS キー、[インポートされたキーマテリアル](#)を持つ KMS キー、または[カスタムキーストア](#)内の KMS キーを自動的にローテーションすることはできません。ただし、[手動でローテーション](#)することはできます。

6. [自動キーローテーション] セクションで、[編集] を選択します。
7. [キーローテーション] で、[有効化] を選択します。

Note

KMS キーが無効になっているか、削除が保留中の場合、AWS KMS はキーマテリアルをローテーションせず、自動キーローテーションステータスまたはローテーション期間を更新することはできません。KMS キーを有効にするか、削除をキャンセルして自動

キーローテーション設定を更新します。詳細については、「[キーローテーションの仕組み](#)」および「[キーの AWS KMS キーステータス](#)」を参照してください。

8. (オプション) 90 日から 2560 日の範囲でローテーション期間を入力します。デフォルト値は 365 日です。カスタムローテーション期間を指定しない場合、AWS KMS はキーマテリアルを毎年ローテーションします。

[kms:RotationPeriodInDays](#) 条件キーを使用して、プリンシパルがローテーション期間に指定できる値を制限することができます。

9. [保存] を選択します。

AWS KMS API の使用

任意のカスタマーマネージドキーについて、[AWS Key Management Service \(AWS KMS\) API](#) を使用して自動キーローテーションを有効化し、現在のローテーションステータスを確認することができます。以下の例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

[EnableKeyRotation](#) オペレーションでは、指定された KMS キーの自動キーローテーションが有効になります。このオペレーションで KMS キーを特定するには、[キー ID](#) または [キー ARN](#) を使用します。デフォルトでは、カスタマーマネージドキーのキーローテーションは無効になっています。

[kms:RotationPeriodInDays](#) 条件キーを使用して、プリンシパルが [EnableKeyRotation](#) リクエストの [RotationPeriodInDays](#) パラメータに指定できる値を制限することができます。

以下の例では、指定された対称暗号化 KMS キーについてローテーション期間を 180 日に設定してキーローテーションを有効にし、[GetKeyRotationStatus](#) オペレーションを使用して結果を確認しています。

```
$ aws kms enable-key-rotation \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --rotation-period-in-days 180

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyRotationEnabled": true,
  "RotationPeriodInDays": 180,
  "NextRotationDate": "2024-02-14T18:14:33.587000+00:00"
}
```

自動キーローテーションを無効にする

カスタマーマネージドキーの自動キーローテーションは、有効化した後にいつでも無効化することができます。

自動キーローテーションを無効化すると、KMS キーは、ローテーションが無効化された時点で使用していたバージョンのキーマテリアルをそのまま使用し続けます。自動キーローテーションを再度有効にすると、は新しいローテーション有効日に基づいてキーマテリアルを AWS KMS ローテーションします。

自動ローテーションを無効化にしても、[オンデマンドローテーション](#)を実行する機能には影響せず、進行中のオンデマンドローテーションもキャンセルされません。

AWS KMS コンソールまたは [DisableKeyRotation](#) オペレーションを使用して、自動キーローテーションを無効にすることができます。自動キーローテーションを無効にするには、kms:DisableKeyRotation アクセス許可が必要です。アクセス AWS KMS 許可の詳細については、「」を参照してください[アクセス許可に関するリファレンス](#)。

AWS KMS コンソールの使用

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[Customer managed keys] (カスタマーマネージドキー) を選択します。(AWS マネージドキーのローテーションを有効化または無効化することはできません。これらのローテーションは毎年自動的に行われます。)
4. KMS キーのエイリアスまたは キー ID を選択します。
5. [キーローテーション] タブを選択します。

[Key rotation] (キーローテーション) タブは、[マルチリージョン](#)の対称暗号化 KMS キーなど、AWS KMS が生成したキーマテリアルを持つ対称暗号化 KMS キー ([Origin] (オリジン) が [AWS_KMS]) の詳細ページにのみ表示されます。

非対称 KMS キー、HMAC KMS キー、[インポートされたキーマテリアル](#)を持つ KMS キー、または[カスタムキーストア](#)内の KMS キーを自動的にローテーションすることはできません。ただし、[手動でローテーション](#)することはできます。

6. [自動キーローテーション] セクションで、[編集] を選択します。
7. [キーローテーション] で、[無効化] を選択します。

Note

KMS キーが無効になっているか、削除が保留中の場合、AWS KMS はキーマテリアルをローテーションせず、自動キーローテーションステータスまたはローテーション期間を更新することはできません。KMS キーを有効にするか、削除をキャンセルして自動キーローテーション設定を更新します。詳細については、「[キーローテーションの仕組み](#)」および「[キーの AWS KMS キーステータス](#)」を参照してください。

8. [保存] を選択します。

AWS KMS API の使用

任意のカスタマーマネージドキーについて、[AWS Key Management Service \(AWS KMS\) API](#) を使用して自動キーローテーションを無効化し、現在のローテーションステータスを確認することができます。この例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用することができます。

[DisableKeyRotation](#) オペレーションは、自動キーローテーションを無効にします。このオペレーションで KMS キーを特定するには、[キー ID](#) または [キー ARN](#) を使用します。デフォルトでは、カスタマーマネージドキーのキーローテーションは無効になっています。

以下の例では、指定された対称暗号化 KMS キーについて自動キーローテーションを無効にし、[GetKeyRotationStatus](#) オペレーションを使用して結果を確認しています。

```
$ aws kms disable-key-rotation --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyRotationEnabled": false
}
```

オンデマンドキーローテーションの実行

自動キーローテーションが有効かどうかにかかわらず、カスタマーマネージド KMS キーのキーマテリアルのオンデマンドローテーションを実行できます。自動ローテーション ([DisableKeyRotation](#)) を無効にしても、オンデマンドローテーションを実行する機能には影響せず、進行中のオンデマン

ドローテーションもキャンセルされません。オンデマンドドローテーションは、既存の自動ドローテーションスケジュールには影響を与えません。例えば、ドローテーション期間を 730 日に設定して自動ドローテーションを有効化している KMS キーを想定してみましょう。キーの自動ドローテーションを 2024 年 4 月 14 日に予定して、オンデマンドドローテーションを 2024 年 4 月 10 日に実行しても、キーはスケジュールどおりに、2024 年 4 月 14 日と以降 730 日ごとに自動ドローテーションされます。

オンデマンドドローテーションは、1 つの KMS キーごとに最大 10 回実行できます。AWS KMS コンソールを使用して、KMS キーで利用できる残りのオンデマンドドローテーションの数を表示できます。

オンデマンドドローテーションは、[対称暗号化 KMS キー](#)でのみサポートされています。[非対称 KMS キー](#)、[HMAC KMS キー](#)、[インポートされたキーマテリアル](#)を持つマルチリージョン KMS キー、または[カスタムキーストア](#)内の KMS キーのオンデマンドドローテーションを実行することはできません。関連する[マルチリージョンキー](#)セットのオンデマンドドローテーションを実行するには、プライマリキーでオンデマンドドローテーションを呼び出します。

kms:RotateKeyOnDemand および アクセスkms:GetKeyRotationStatus許可を持つ承認済みユーザーは、AWS KMS コンソールと AWS KMS API を使用してオンデマンドドローテーションを開始し、ドローテーションのステータスを表示できます。[ListKeyRotations](#) を使用して、KMS キーの完了済みのドローテーションを表示します。

トピック

- [オンデマンドドローテーションの開始 \(コンソール\)](#)
- [オンデマンドドローテーションの開始 \(AWS KMS API\)](#)

オンデマンドドローテーションの開始 (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。(AWS マネージドキーをオンデマンドドローテーションすることはできません。これらのキーのドローテーションは毎年自動的に行われます。)
4. KMS キーのエイリアスまたは キー ID を選択します。
5. キーマテリアルとドローテーションタブを選択します。

キーマテリアルとローテーションタブは、自動ローテーションまたはオンデマンドローテーションをサポートする対称暗号化 KMS キーの詳細ページにのみ表示されます。これには、[が AWS KMS 生成したキーマテリアルを持つ KMS キー \(AWS_KMS オリジン\)](#) と、[インポートされたキーマテリアルを持つ単一リージョン KMS キー \(EXTERNAL オリジン\)](#) が含まれます。

非対称 KMS キー、HMAC KMS キー、[インポートされたキーマテリアルを持つマルチリージョン KMS キー](#)、または[カスタムキーストアの KMS キー](#)のオンデマンドローテーションを実行することはできません。ただし、[手動でローテーション](#)することはできます。

6. [今すぐ回転](#) を選択します。インポートされたキーマテリアルを持つ単一リージョンの対称暗号化キーの場合、Rotate now オプションは、以前に[新しいキーマテリアルをインポート](#)し、それが保留中のローテーション状態である場合にのみ使用できます。
7. そのキーの残りのオンデマンドローテーション回数に関する注意事項が表示されるので、確認します。また、ローテーション後に最新になるキーマテリアルの ID、説明、有効期限などの情報も表示されます。オンデマンドローテーションを続行しない場合は、[キャンセル] を選択します。
8. [キーをローテーションする] を選択して、オンデマンドローテーションの実行を確定します。

 Note

オンデマンドローテーションは、他の AWS KMS 管理オペレーションと同じ結果整合性の影響を受けます。新しいキーマテリアルが AWS KMS 全体で使用可能になるまで、若干の遅延が生じることがあります。オンデマンドローテーションが完了すると、コンソールの上部にあるバナーに通知が表示されます。

オンデマンドキーローテーションの開始 (AWS KMS API)

任意のカスタマーマネージドキーについて、[AWS Key Management Service \(AWS KMS\) API](#) を使用してオンデマンドキーローテーションを開始し、現在のローテーションステータスを確認することができます。この例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用することができます。

[RotateKeyOnDemand](#) オペレーションは、指定された KMS キーのオンデマンドキーローテーションを直ちに開始します。これらのオペレーションで KMS キーを識別するには、その[キー ID](#) または[キー ARN](#) を使用します。

次の例では、指定された対称暗号化 KMS キーについてオンデマンドキーローテーションを開始し、[GetKeyRotationStatus](#) オペレーションを使用してオンデマンドローテーションが進行中であることを確認しています。kms:GetKeyRotationStatus レスポンス内の OnDemandRotationStartDate は、進行中のオンデマンドローテーションが開始された日時を特定します。この例では、KMS キーで自動ローテーションも有効になっており、期間は 365 日間です。

```
$ aws kms rotate-key-on-demand --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyRotationEnabled": true,
  "NextRotationDate": "2024-03-14T18:14:33.587000+00:00",
  "OnDemandRotationStartDate": "2024-02-24T18:44:48.587000+00:00"
  "RotationPeriodInDays": 365
}
```

KMS キーが自動ローテーションをサポートしていない場合、または自動ローテーションが有効になっていない場合、次の例に示すようにkms:GetKeyRotationStatusレスポンスのフィールドが少なくなります。

```
$ aws kms rotate-key-on-demand --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}

$ aws kms get-key-rotation-status --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyRotationEnabled": false,
  "OnDemandRotationStartDate": "2024-02-24T18:44:48.587000+00:00"
}
```

ローテーションとキーマテリアルを一覧表示する

自動キーローテーションまたはオンデマンドキーローテーションをサポートする KMS キーには、複数のキーマテリアルを関連付けることができます。これらのキーには、自動ローテーションまたはオンデマンドローテーションごとに初期キーマテリアルと 1 つの追加キーマテリアルがあります。

kms:ListKeyRotations 権限を持つ承認済みユーザーは、AWS KMS コンソールと [ListKeyRotations](#) API を使用して、完了した自動ローテーションとオンデマンドローテーションからのものを含め、KMS キーに関連付けられたすべてのキーマテリアルを一覧表示できます。

トピック

- [ローテーションとキーマテリアルを一覧表示する \(コンソール\)](#)
- [ローテーションとキーマテリアルを一覧表示する \(AWS KMS API\)](#)

ローテーションとキーマテリアルを一覧表示する (コンソール)

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
 2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
 3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
 4. KMS キーのエイリアスまたは キー ID を選択します。
 5. キーマテリアルとローテーションタブを選択します。
- キーマテリアルとローテーションタブは、自動ローテーションまたはオンデマンドローテーションをサポートする対称暗号化 KMS キーの詳細ページにのみ表示されます。これには、(AWS_KMS オリジン) AWS KMS を生成したキーマテリアルを持つ KMS キーと、インポートされたキーマテリアル (EXTERNAL オリジン) を持つ単一リージョン KMS キーが含まれます。
 - キーマテリアルとローテーションタブのキーマテリアルテーブルには、KMS キーに関連付けられているすべてのキーマテリアルが一覧表示されます。キーマテリアルごとに、対応するエントリには、によって割り当てられた一意の識別子 AWS KMS、ローテーション日、およびキーマテリアルの状態が表示されます。ローテーション日は、自動またはオンデマンドのキーローテーション後にキーマテリアルがいつ最新になったかを識別します。最初のマテリアルまたは Pending rotation キーマテリアルに関連付けられたローテーション日はありません。キーマテリアルの状態によって、がキーマテリアル AWS KMS を使用する方法が決まります。現在のキーマテリアルは、暗号化と復号の両方に使用されます。最新でないキーマテリ

アルは復号にのみ使用されます。キーマテリアルの状態は、キーマテリアルがローテーションのためにステージングされているPending rotationことを示します。このキーマテリアルは、オンデマンドキーローテーションによって現在のキーマテリアルになるまで、暗号化オペレーションには使用されません。キーマテリアルに表示される追加情報は、KMS キーのタイプによって異なります。

- オAWS_KMSリジンを持つ対称暗号化 KMS キーの場合、各行にはローテーションタイプ - On-demandまたは も表示されますAutomatic。
- インポートされたキーマテリアル (EXTERNAL オリジン) を持つ単一リージョンの対称暗号化 KMS キーはローOn-demandテーションのみをサポートするため、ローテーションタイプの列はありません。代わりに、各行にはインポート状態、ユーザーが指定した説明、有効期限情報、アクションメニューが表示されます。インポート状態は、キーマテリアルが内部で使用可能であることを示すインポート、AWS KMS またはキーマテリアルが内部で使用可能でないことを示すインポート保留中のいずれかです AWS KMS。Actions メニューを使用して、インポートされたキーマテリアルを削除したり、キーマテリアルを再インポートしたりできます。キーマテリアルの削除アクションは、キーマテリアルのインポート状態がインポート保留中の場合、無効になります。キーマテリアルの再インポートアクションは常に使用できます。再インポートする前に、キーマテリアルの有効期限が切れたり削除されたりするのを待つ必要はありません。

ローテーションとキーマテリアルを一覧表示する (AWS KMS API)

[AWS Key Management Service \(AWS KMS\) API](#) を使用して、オンデマンドキーローテーションを開始し、カスタマーマネージドキーの現在のローテーションステータスを表示できます。この例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用することができます。

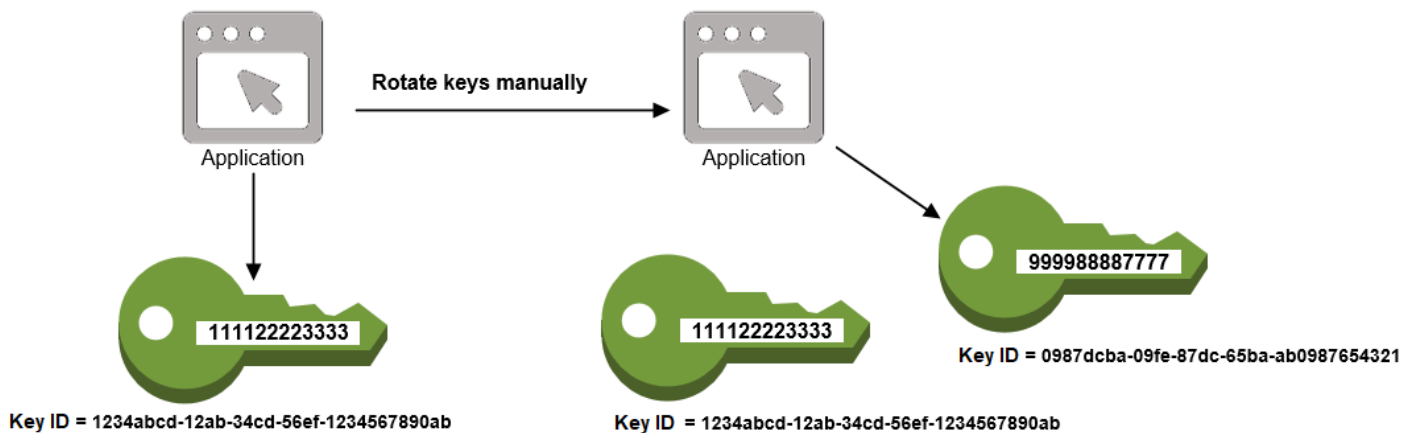
[ListKeyRotations](#) オペレーションは、指定された KMS キーのすべてのローテーションとキーマテリアルを一覧表示します。これらのオペレーションで KMS キーを識別するには、その[キー ID](#) または[キー ARN](#) を使用します。

このオペレーションでは、オプションの IncludeKeyMaterial パラメータがサポートされています。このパラメータのデフォルト値は ROTATIONS_ONLY です。このパラメータを省略すると、は自動またはオンデマンドのキーローテーションによって作成されたキーマテリアルに関する情報 AWS KMS を返します。の値を指定するとALL_KEY_MATERIAL、 は最初のキーマテリアルとローテーション保留中のインポートされたキーマテリアルをレスポンス AWS KMS に追加します。このパラメータは、自動キーローテーションまたはオンデマンドキーローテーションをサポートする KMS キーでのみ使用できます。

```
$ aws kms list-key-rotations --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --include-key-material ALL_KEY_MATERIAL
{
  "Rotations": [
    {
      "KeyId": 1234abcd-12ab-34cd-56ef-1234567890ab,
      "KeyMaterialId":
123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0,
      "KeyMaterialDescription": "KeyMaterialA",
      "ImportState": "PENDING_IMPORT",
      "KeyMaterialState": "NON_CURRENT"
    },
    {
      "KeyId": 1234abcd-12ab-34cd-56ef-1234567890ab,
      "KeyMaterialId":
96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068,
      "ImportState": "IMPORTED",
      "KeyMaterialState": "CURRENT",
      "ExpirationModel": "KEY_MATERIAL_DOES_NOT_EXPIRE",
      "RotationDate": "2025-05-01T15:50:51.045000-07:00",
      "RotationType": "ON_DEMAND"
    }
  ],
  "Truncated": false
}
```

キーを手動でローテーションする

自動キーローテーションやオンデマンドキーローテーションを使用する代わりに、新しい KMS キーを作成し、現在の KMS キーの代わりに使用することもできます。新しい KMS キーが現在の KMS キーと異なる暗号化マテリアルを持つ場合、新しい KMS キーを使用すると、既存の KMS キーでキーマテリアルを変更するのと同じ効果があります。1 つの KMS キーを別のものと置き換えるプロセスは、手動キーローテーションと呼ばれます。



手動ローテーションは、非対称 KMS キー、HMAC KMS キー、[カスタムキーストア](#)の KMS キー、インポートされたキー[マテリアル](#)を持つマルチリージョン KMS キーなど、自動キーローテーションまたはオンデマンドキーローテーションの対象ではない KMS キーをローテーションする場合に適しています。

Note

新しい KMS キーの使用を開始するときは、元の KMS キーを有効にしたままにして、が元の KMS キーが暗号化したデータを復号 AWS KMS できるようにします。

KMS キーを手動で更新すると、アプリケーションの KMS キー ID または キーの ARN へのリファレンスも更新する必要があります。KMS キーにわかりやすい名前を関連付けられる[エイリアス](#)が、このプロセスを容易にします。エイリアスを使用して、アプリケーションの KMS キーを参照します。続いて、アプリケーションが使用する KMS キーを変更するには、アプリケーションのコードを編集する代わりに、エイリアスのターゲット KMS キーを変更します。詳細については、「[アプリケーションでエイリアスを使用する方法について説明します。](#)」を参照してください。

Note

手動でローテーションされた KMS キーの最新バージョンを指すエイリアスは、[DescribeKey](#)、[GetPublicKey](#)、および [DeriveSharedSecret](#)、[Encrypt](#)、[GenerateDataKey](#)、[GenerateDataKeyPair](#)、[GenerateMac](#)、[VerifyMac](#) and [Verify](#) などの暗号化オペレーションに適しています。[DisableKey](#) または [ScheduleKeyDeletion](#) のような、KMS キーを管理するオペレーションでは、エイリアスは許可されていません。

手動でローテーションされた対称暗号化 KMS キーで [Decrypt](#) オペレーションを呼び出す場合、コマンドから `KeyId` パラメータを省略します。は暗号文を暗号化した KMS キー AWS KMS を自動的に使用します。

非対称 KMS キーで `Decrypt` または [Verify](#) を呼び出す場合や、HMAC KMS キーで [VerifyMac](#) を呼び出す場合は、`KeyId` パラメータが必要です。キーが手動で更新された場合など、暗号化オペレーションを実行する KMS キーをもはや指していないエイリアス `KeyId` パラメータの値になっている場合には、これらのリクエストは失敗します。このエラーを回避するには、オペレーションごとに正しい KMS キーを追跡して、指定する必要があります。

エイリアスのターゲット KMS キーを変更するには、[UpdateAlias](#) オペレーションを AWS KMS API で使用します。例えば、このコマンドでは、新しい KMS キーを指すように `alias/TestKey` エイリアスを更新します。オペレーションは出力を返さないため、この例では [ListAliases](#) オペレーションを使用して、エイリアスが異なる KMS キーと関連付けられ、`LastUpdatedDate` フィールドが更新されたことを示します。ListAliases コマンドは、の `alias/TestKey` [query パラメータ](#) を使用してエイリアスのみ AWS CLI を取得します。

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/TestKey`]'
{
  "Aliases": [
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestKey",
      "AliasName": "alias/TestKey",
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "CreationDate": 1521097200.123,
      "LastUpdatedDate": 1521097200.123
    },
  ]
}

$ aws kms update-alias --alias-name alias/TestKey --target-key-id
0987dcba-09fe-87dc-65ba-ab0987654321

$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/TestKey`]'
{
  "Aliases": [
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/TestKey",
```

```
    "AliasName": "alias/TestKey",
    "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "CreationDate": 1521097200.123,
    "LastUpdatedDate": 1604958290.722
  },
]
```

マルチリージョンキーセットのプライマリキーを変更する

関連するマルチリージョンのキーのセットはすべて、プライマリキーを持つ必要があります。ただし、プライマリキーは変更できます。このアクションは、プライマリリージョンの更新と呼ばれ、現在のプライマリキーをレプリカキーに変換し、関連するレプリカキーの1つをプライマリキーに変換します。これは、レプリカキーを維持しながら現在のプライマリキーを削除する必要がある場合、またはキー管理者と同じリージョンでプライマリキーを検索する必要がある場合に実行できます。

関連する任意のレプリカキーを選択して、新しいプライマリキーにすることができます。オペレーションのスタート時に、プライマリキーとレプリカキーの両方が Enabled [キーステータス](#) である必要があります。

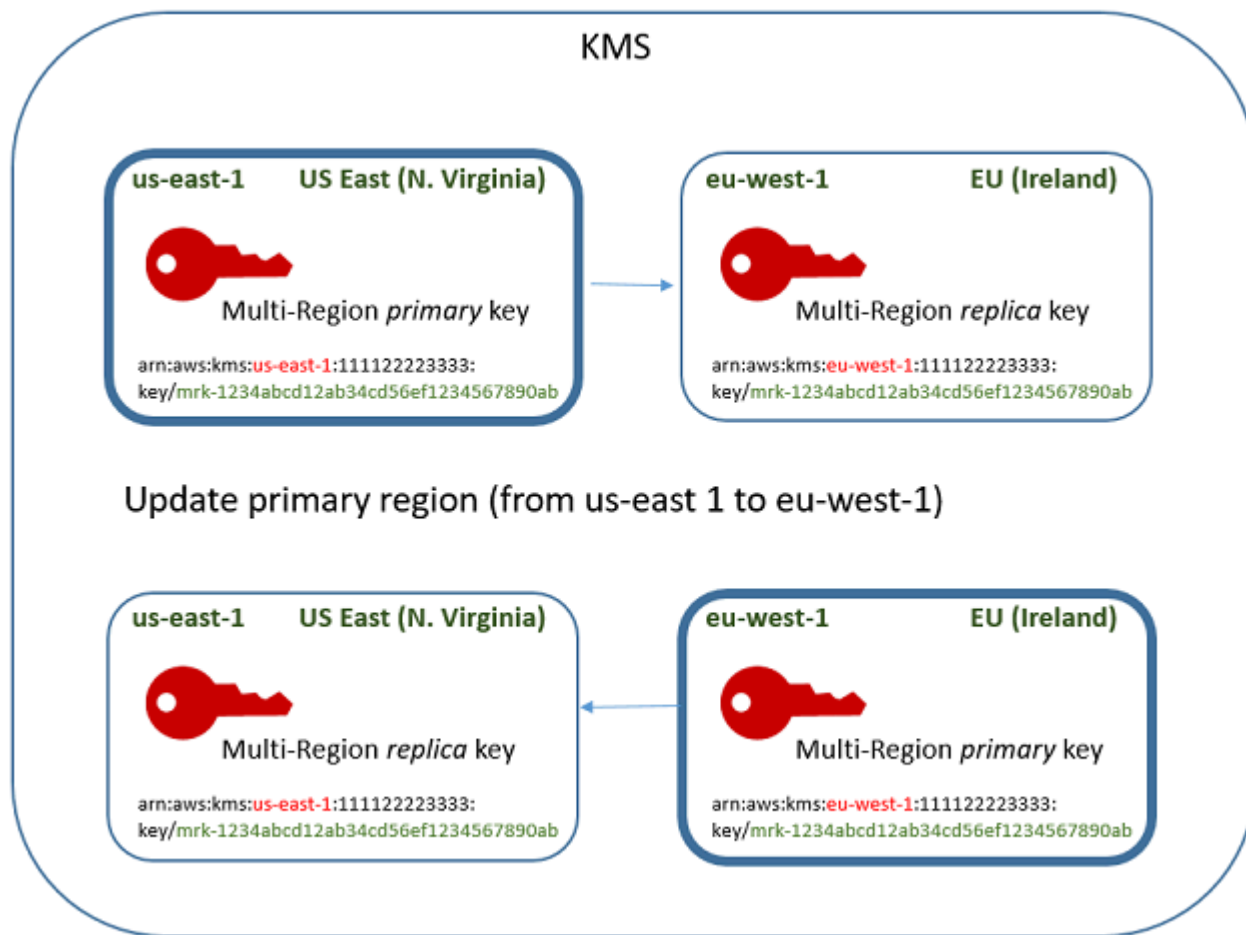
Updating キーステータス

UpdatePrimaryRegion オペレーション完了後も、プライマリリージョンの更新プロセス完了までにさらに数秒間かかる可能性があります。この間、新旧プライマリキーのキーステータスは、一時的に [更新中](#) となります。キーステータスが Updating の間も暗号化オペレーションでキーを使用できますが、新しいプライマリキーをレプリケートしたり、これらのキーを有効または無効にするなどの、特定の管理オペレーションを実行することはできません。[DescribeKey](#) などのオペレーションは、新旧のプライマリキー両方をレプリカとして表示することがあります。更新が完了すると、Enabled キーステータスは復元されます。

Updating キーステータスの影響の詳細については、[キーの AWS KMS キーステータス](#) を参照してください。

仕組み

米国東部 (バージニア北部) (us-east-1) にプライマリキーがあり、欧州 (アイルランド) (eu-west-1) にレプリカキーがあるとします。更新機能を使用して、米国東部 (バージニア北部) (us-east-1) のプライマリキーをレプリカキーに変更し、欧州 (アイルランド) (eu-west-1) のレプリカキーをプライマリキーに変更できます。



更新プロセスが完了すると、欧州 (アイルランド) (eu-west-1) リージョンのマルチリージョンキーがマルチリージョンのプライマリーキーになり、米国東部 (バージニア北部) (us-east-1) リージョンのキーがそのレプリカキーになります。関連するレプリカキーが他に存在する場合、それらは新しいプライマリーキーのレプリカになります。が次にマルチリージョンキーの共有プロパティを AWS KMS 同期すると、新しいプライマリーキーから [共有プロパティ](#) を取得し、以前のプライマリーキーを含むレプリカキーにコピーします。

更新オペレーションは、任意のマルチリージョンキーの [キー ARN](#) には影響を及ぼしません。また、キーマテリアルなどの共有プロパティや、キーポリシーなどの独立したプロパティにも影響を及ぼしません。ただし、新しいプライマリーキーの [キーポリシーの更新](#) が必要になります。例えば、信頼できるプリンシパルの [kms:ReplicateKey](#) アクセス許可を新しいプライマリーキーに追加し、新しいレプリカキーから削除します。

プライマリリージョンを更新する

レプリカキーをプライマリキーに変換することができます。変換後、元のプライマリキーはレプリカキーに変化します。プライマリリージョンを更新するには、両方のリージョンでの [kms:UpdatePrimaryRegion](#) 許可が必要です。

プライマリリージョンは、AWS KMS コンソールで更新するか、[UpdatePrimaryRegion](#) オペレーションを使用して更新できます。

AWS KMS コンソールの使用

AWS KMS コンソールでプライマリキーを更新できます。現在のプライマリキーのキーの詳細ページをスタートします。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. [マルチリージョンのプライマリキー](#) のキー ID またはエイリアスを選択します。これにより、プライマリキーのキーの詳細ページが開きます。

マルチリージョンのプライマリキーを識別するには、右上隅にあるツールアイコンを使用して [Regionality] (リージョナリティー) 列をテーブルに追加します。

5. [Regionality] (リージョナリティー) タブを選択します。
6. [Primary key (プライマリキー)] セクションで、[Change primary Region (プライマリリージョンの変更)] を選択します
7. 新しいプライマリキーのリージョンを選択します。メニューから選択できるリージョンは 1 つだけです。

[Change primary Region (プライマリリージョンの変更)] メニューには、関連するマルチリージョンキーを持つリージョンのみが含まれます。メニュー上のすべてのリージョンに対する [プライマリリージョンを更新するアクセス許可](#) は持っていない可能性があります。

8. [Change primary Region (プライマリリージョンの変更)] を選択します。

AWS KMS API の使用

関連するマルチリージョンキーのセットでプライマリキーを変更するには、[UpdatePrimaryRegion](#) オペレーションを使用します。

KeyId パラメータを使用して、現在のプライマリキーを識別します。PrimaryRegion パラメータを使用して、新しいプライマリキー AWS リージョン の を指定します。プライマリキーに新しいプライマリリージョンのレプリカがまだない場合、オペレーションは失敗します。

次の例では、プライマリキーを us-west-2 リージョンのマルチリージョンキーから eu-west-1 リージョンのレプリカに変更します。KeyId パラメータは、現在のプライマリキーを us-west-2 リージョンで識別します。PrimaryRegion パラメータは、新しいプライマリキー AWS リージョン の を指定します eu-west-1。

```
$ aws kms update-primary-region \
    --key-id arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \
    --primary-region eu-west-1
```

成功すると、このオペレーションは出力を返さず、HTTP ステータスコードのみを返します。効果を確認するには、[DescribeKey](#) オペレーションをマルチリージョンキーのいずれかで呼び出します。キーステータスが Enabled に戻るまで待機する必要がある場合があります。キーステータスが [更新中](#)の間は、キー値がまだ流動的である可能性があります。

例えば、次の DescribeKey 呼び出しにより、eu-west-1 リージョンでマルチリージョンキーの詳細を取得します。出力は、eu-west-1 リージョンのマルチリージョンキーが現在、プライマリキーであることを示します。関連する us-west-2 リージョンのマルチリージョンキー (同じキー ID) は現在、レプリカキーです。

```
$ aws kms describe-key \
    --key-id arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab \

{
  "KeyMetadata": {
    "AWSAccountId": "111122223333",
    "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
    "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
    "CreationDate": 1609193147.831,
    "Enabled": true,
    "Description": "multi-region-key",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "Origin": "AWS_KMS",
```

```
    "KeyManager": "CUSTOMER",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
    ],
    "MultiRegion": true,
    "MultiRegionConfiguration": {
        "MultiRegionKeyType": "PRIMARY",
        "PrimaryKey": {
            "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
            "Region": "eu-west-1"
        },
        "ReplicaKeys": [
            {
                "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "us-west-2"
            }
        ]
    }
}
```

を削除する AWS KMS key

の削除 AWS KMS key は破壊的であり、潜在的に危険です。これは、キーマテリアルと KMS キーに関連付けられているすべてのメタデータを削除し、元に戻すことはできません。KMS キーを削除すると、その KMS キーで暗号化されたデータを復号できなくなります。これは、そのデータが回復不能になることを意味します。(唯一の例外は、[マルチリージョンのレプリカキー](#)と、キーマテリアルを含む非対称キーと HMAC KMS キーです。) このリスクは、[暗号化に使用される非対称 KMS キー](#)で、警告やエラーなしに、プライベートキーの削除後に復号できないパブリックキーを含む暗号文をユーザーが引き続き生成できる場合に重要です AWS KMS。

KMS キーの削除は、そのキーをもう使用しないことが確実である場合にのみ行ってください。不明な場合は、削除するのではなく、[KMS キーを無効化する](#)ことを検討します。無効にした KMS キーを再度有効にしたり、KMS キーの[削除のスケジュールをキャンセル](#)したりすることは可能ですが、すでに削除した KMS キーを復元することはできません。

スケジュールできるのは、カスタマーマネージドキーの削除のみです。AWS マネージドキー または は削除できません AWS 所有のキー。

KMS キーを削除する前に、その KMS キーで暗号化された暗号文の数を知りたい場合があります。AWS KMS はこの情報を保存せず、暗号文も保存しません。この情報を取得するには、KMS キーの過去の使用状況を特定する必要があります。ヘルプについては、[KMS キーの過去の使用状況を確認する](#)を参照してください。

AWS KMS 削除を明示的にスケジュールし、必須の待機期間が終了しない限り、 は KMS キーを削除しません。

KMS キーを削除する理由には次の 1 つ以上のものが考えられます。

- 不要になった KMS キーのキーライフサイクルを完了する
- 使用しない KMS キーの維持に伴う管理オーバーヘッドと[コスト](#)を回避する
- [KMS キーリソースクォータ](#)に対してカウントされる KMS キーの数を減らすには

Note

[を閉じる AWS アカウント](#)と、KMS キーにアクセスできなくなり、課金されなくなります。

AWS KMS は、KMS キーの[削除をスケジュール](#)したとき、および [KMS キーが実際に削除](#)されたときに AWS CloudTrail、ログにエントリを記録します。

待機期間について

KMS キーの削除は破壊的で潜在的に危険であるため、では 7～30 日間の待機期間を設定 AWS KMS する必要があります。デフォルトの待機時間は、30 日です。

ただし、実際の待機期間は、スケジュールした待機期間よりも最大 24 時間長くなる場合があります。KMS キーが削除される実際の日付と時刻を取得するには、[DescribeKey](#) オペレーションを使用します。または、AWS KMS コンソール、KMS キーの[詳細ページ](#)、[General configuration] (一般的な設定) セクションで、スケジュールされた削除の日付を参照してください。必ずタイムゾーンをメモしておきます。

削除の待機期間中は、KMS キーステータスおよびキーの状態が削除保留中になります。

- 削除保留中の KMS キーを[暗号化オペレーション](#)で使用することはできません。
- AWS KMS は、[削除保留中の KMS キーのキーマテリアルをローテーション](#)しません。

待機期間が終了すると、は KMS キー、エイリアス、および関連するすべての AWS KMS メタデータ AWS KMS を削除します。

KMS キーの削除をスケジュールしても、KMS キーで暗号化されたデータキーに、ただちに影響しない場合もあります。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

待機期間を設定することにより、KMS キーが不要であり、今後も使用することがないことを確認できます。待機期間中にユーザーまたはアプリケーションが KMS キーの使用を試みた場合に警告するように [Amazon CloudWatch アラームを設定](#)できます。KMS キーを復元するには、待機期間の終了前にキーの削除をキャンセルします。待機期間が終了すると、はキーの削除をキャンセルできず、は KMS キー AWS KMS を削除します。

特別な考慮事項

キー削除をスケジュールする前に、特定用途向けの KMS キーの削除に関する以下の注意事項を確認してください。

非対称 KMS キーの削除

[認可されたユーザー](#)は、対称または非対称 KMS キーを削除できます。これらの KMS キーの削除をスケジュールする手順は、どちらの種類のキーも同じです。ただし、[非対称 KMS キーのパブリックキーはの外部でダウンロード](#)して使用できるため AWS KMS、特に暗号化に使用される非対称 KMS キー (キーの使用状況は) の場合、オペレーションには大きなリスクが伴います ENCRYPT_DECRYPT。

- KMS キーの削除をスケジュールすると、KMS キーのキーステータスが削除保留中に変わり、KMS キーを[暗号化オペレーション](#)で使えなくなります。ただし、削除のスケジュールは 外のパブリックキーには影響しません AWS KMS。パブリックキーを持つユーザーは、引き続きそのパブリックキーを使ってメッセージを暗号化できます。キーの状態が変更されたという通知は受信しません。削除がキャンセルされない限り、パブリックキーで作成された暗号文は復号できません。
- 削除保留中の KMS キーを使用する試みを検出するアラーム、ログ、その他の戦略では、AWS KMSの外部でのパブリックキーの使用は検出できません。
- KMS キーが削除されると、その KMS キーを含むすべての AWS KMS アクションは失敗します。ただし、パブリックキーを持つユーザーは、引き続きそのパブリックキーを使ってメッセージを暗号化できます。これらの暗号文は復号できません。

キーの使用方法が ENCRYPT_DECRYPT である非対称 KMS キーを削除する必要がある場合は、CloudTrail ログエントリを使用して、パブリックキーがダウンロードおよび共有されているかどうかを確認します。完了している場合は、パブリックキーが AWS KMSの外部で使用されていないことを確認します。次に、削除するのではなく、[KMS キーを無効にする](#)ことを検討します。

非対称 KMS キーの削除によって生じるリスクは、インポートされたキーマテリアルを含む非対称 KMS キーであれば軽減されます。 詳細については、「[Deleting KMS keys with imported key material](#)」を参照してください。

マルチリージョンキーの削除

プライマリキーを削除するには、すべてのレプリカキーの削除をスケジュールし、レプリカキーが削除されるまで待機する必要があります。プライマリキーの削除に必要な待機時間は、最後のレプリカキーが削除された時点から始まります。レプリカキーを削除せずに特定のリージョンからプライマリキーを削除する必要がある場合は、[プライマリリージョンの更新](#)を使用して、プライマリキーをレプリカキーに変更します。

レプリカキーはいつでも削除することができます。これは、他の KMS キーのキーステータスに依存しません。誤ってレプリカキーを削除した場合は、同じリージョンで同じプライマリキーを

レプリケートすることで再度作成できます。作成した新しいレプリカキーは、元のレプリカキーと同じ[共有プロパティ](#)を有します。

インポートされたキーマテリアルを含む KMS キーの削除

キーマテリアルがインポートされた KMS キーのキーマテリアルの削除は一時的で、元に戻すことができます。キーを復元するには、キーマテリアルを再インポートします。

一方、KMS キーの削除は破棄できません。[キーの削除をスケジュール](#)し、必要な待機期間が終了すると、は KMS キー、そのキーマテリアル、および KMS キーに関連付けられたすべてのメタデータを AWS KMS 永続的および元に戻すことなく削除します。

ただし、キーマテリアルがインポートされた KMS キーを削除した場合のリスクと結果は、KMS キーのタイプ (「キー仕様」) によって異なります。

- 対称暗号化キー – 対称暗号化 KMS キーを削除すると、そのキーで暗号化された残りの暗号文はすべて回復できなくなります。同じキーマテリアルを使用しても、削除された対称暗号化 KMS キーの暗号文を復号できる新しい対称暗号化 KMS キーを作成することはできません。各 KMS キーに固有のメタデータは、各対称暗号文に暗号的にバインドされます。このセキュリティ機能により、対称暗号文を暗号化した KMS キーのみで復号できることが保証されますが、同等の KMS キーを再作成することができなくなっています。
- 非対称キーと HMAC キー – 元のキーマテリアルがある場合は、削除された非対称キーまたは HMAC KMS キーと同じ暗号化プロパティを持つ新しい KMS キーを作成できます。は、一意のセキュリティ機能を含まない標準の RSA 暗号文と署名、ECC 署名、および HMAC タグ AWS KMS を生成します。また、HMAC キーまたは非対称キーペアのプライベートキーを AWS の外部で使用できます。

同じ非対称キーマテリアルまたは HMAC キーマテリアルを使用して作成した新しい KMS キーには、異なるキー識別子が割り当てられます。新しいキーポリシーを作成し、エイリアスをすべて再作成し、新しいキーを参照するように既存の IAM ポリシーとアクセス許可を更新する必要があります。

キーストアからの KMS AWS CloudHSM キーの削除

AWS CloudHSM キーストアからの KMS キーの削除をスケジュールすると、[そのキーの状態](#)は削除保留中に変わります。KMS キーは、[カスタムキーストアの切断](#)によって KMS キーが使用できなくなった場合でも、待機期間中を通して削除保留中ステータスを維持します。これにより、待機期間中はいつでも KMS キーの削除をキャンセルできます。

待機期間が終了すると、は KMS キー AWS KMS を削除します AWS KMS。次に AWS KMS、は、関連付けられた AWS CloudHSM クラスタからキーマテリアルを削除するために最善を尽

くします。キーストアが AWS KMS から切断されるなど、AWS KMS でキーマテリアルを削除できない場合は、クラスターから手動で[孤立したキーマテリアルを削除](#)できます。

AWS KMS はクラスターバックアップからキーマテリアルを削除しません。から KMS キーを削除 AWS KMS し、そのキーマテリアルを AWS CloudHSM クラスターから削除した場合でも、バックアップから作成されたクラスターには削除されたキーマテリアルが含まれている可能性があります。キーマテリアルを恒久的に削除するには、[DescribeKey](#) オペレーションを使用してその KMS キーの作成日を特定します。次に、キーのマテリアルを含む可能性のある [すべてのクラスターバックアップを削除します](#)。

AWS CloudHSM キーストアからの KMS キーの削除をスケジュールすると、KMS キーはすぐに使用できなくなります (結果整合性の影響を受ける)。ただし、KMS キーで保護された[データキー](#)で暗号化されているリソースは、KMS キーが (データキーの復号などで) 再度使用されるまで、その影響を受けません。この問題は AWS のサービスに影響します。その多くは、データキーを使用してリソースを保護します。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

外部キーストアからの KMS キーの削除

外部キーストアから KMS キーを削除しても、キーマテリアルとして使用されていた[外部キー](#)には影響しません。

外部キーストアの KMS キーの削除をスケジュールすると、[キーステータス](#)が [Pending deletion] (削除保留中) に変わります。KMS キーは、[外部キーストアの切断](#)によって KMS キーが使用できなくなった場合でも、待機期間中を通して [Pending deletion] (削除保留中) ステータスを維持します。これにより、待機期間中はいつでも KMS キーの削除をキャンセルできます。待機期間が終了すると、は KMS キー AWS KMS を削除します AWS KMS。

外部キーストアの KMS キーの削除をスケジュールすると、その KMS キーは直ちに使用できなくなります (結果整合性の影響を受けます)。ただし、KMS キーで保護された[データキー](#)で暗号化されているリソースは、KMS キーが (データキーの復号などで) 再度使用されるまで、その影響を受けません。この問題は AWS のサービスに影響します。その多くが、リソースを保護するためにデータキーを使用しています。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

キー削除へのアクセスを制御する

IAM ポリシーを使用して AWS KMS アクセス許可を許可する場合、管理者アクセス ("Action": "*") または AWS KMS フルアクセス ("Action": "kms:*") を持つ AWS IAM アイデンティティ

は、KMS キーの削除をスケジュールおよびキャンセルすることを既に許可されています。キー管理者がキーポリシーでキーの削除をスケジュールおよびキャンセルできるようにするには、AWS KMS コンソールまたは AWS KMS API を使用します。

通常、キーの削除をスケジュールおよびキャンセルするための、アクセス許可を持っているのは、キー管理者のみです。ただし、これらのアクセス許可は、`kms:ScheduleKeyDeletion` と `kms:CancelKeyDeletion` のアクセス許可をキーポリシーまたは IAM ポリシーに追加すれば、他の IAM ID に付与することができます。[kms:ScheduleKeyDeletionPendingWindowInDays](#) 条件キーを使用して、プリンシパルが [ScheduleKeyDeletion](#) リクエストの `PendingWindowInDays` パラメータに指定できる値をさらに制限することもできます。

キー管理者に、キー削除のスケジュールおよびキャンセルを許可する

AWS KMS コンソールを使用する場合

キー管理者に、キーの削除をスケジュールおよびキャンセルするためのアクセス許可を付与するには。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。
4. アクセス許可を変更する KMS キーのエイリアスまたはキー ID を選択します。
5. [Key policy] (キーポリシー) タブを選択します。
6. 次のステップは、キーポリシーの既定のビューとポリシービューで異なります。既定のビューは、既定のコンソールキーポリシーを使用している場合のみ、使用できます。使用していない場合は、ポリシービューのみを使用できます。

既定のビューを使用できる場合は、[Key policy] (キーポリシー) タブに [Switch to policy view] (ポリシービューに切り替え) または [Switch to default view] (既定のビューに切り替え) のいずれかのボタンが表示されます。

- 既定のビュー:
 - [Key deletion] (キーの削除) で、[Allow key administrators to delete this key] (キー管理者にこのキーの削除を許可する) を選択します。
- ポリシービュー:
 - a. [編集] を選択します。

- b. キー管理者向けのポリシーステートメントで、Action 要素に `kms:ScheduleKeyDeletion` と `kms:CancelKeyDeletion` のアクセス許可を追加します。

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSKeyAdmin"},
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms>Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

- c. [Save changes] (変更の保存) をクリックします。

AWS KMS API の使用

を使用して AWS Command Line Interface、キーの削除をスケジュールおよびキャンセルするためのアクセス許可を追加できます。

キーの削除をスケジュールおよびキャンセルするアクセス許可を追加するには

1. [aws kms get-key-policy](#) コマンドを使用して、既存のキーポリシーを取得し、ポリシードキュメントをファイルに保存します。
2. 任意のテキストエディタでポリシードキュメントを開きます。キー管理者向けのポリシーステートメントで、`kms:ScheduleKeyDeletion` と `kms:CancelKeyDeletion` のアクセス許可を追加します。次の例は、これらの 2 つのアクセス許可を持つポリシーステートメントを示しています。

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSKeyAdmin"},
  "Action": [
    "kms:Create*",
    "kms:Describe*",
    "kms:Enable*",
    "kms:List*",
    "kms:Put*",
    "kms:Update*",
    "kms:Revoke*",
    "kms:Disable*",
    "kms:Get*",
    "kms>Delete*",
    "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

3. [aws kms put-key-policy](#) コマンドを使用して、キーポリシーを KMS キーに適用します。

キー削除をスケジュールする

次の手順では、と AWS KMS API AWS KMS を使用して、でキーの削除をスケジュール AWS Management Console し、AWS KMS keys (KMS キー) のキーの削除をキャンセルする方法について説明します。

Warning

KMS キーを削除することは破壊的であり、リスクを伴います。KMS キーが不要であり、今後もしも使用しないことが確実である場合にのみ実行してください。不明な場合は、削除するのではなく [KMS キーを無効化する](#) べきです。

KMS キーを削除する前に、削除するための許可を取得する必要があります。これらのアクセス許可をキー管理者に付与する方法の詳細は、「[キー削除へのアクセスを制御する](#)」を参照してください

い。 [kms:ScheduleKeyDeletionPendingWindowInDays](#) 条件キーを使用して、最小待機期間を設定するなど、待機期間をさらに制限することもできます。

AWS KMS は、KMS キーの [削除をスケジュール](#) したとき、および [KMS キーが実際に削除](#) されたときに AWS CloudTrail、ログにエントリを記録します。

AWS KMS コンソールを使用する場合

では AWS Management Console、複数の KMS キーの削除を一度にスケジュールおよびキャンセルできます。

キーの削除をスケジュールするには

1. にサインイン AWS Management Console し、 <https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。

[AWS マネージドキー](#) または [AWS 所有のキー](#) の削除をスケジュールすることはできません。

4. 削除する KMS キーの横にあるチェックボックスをオンにします。
5. [Key actions] (キーのアクション)、[Schedule key deletion] (キーの削除をスケジュール) の順に選択します。
6. 待機中に、警告、および削除のキャンセルに関する情報を読み、検討してください。削除をキャンセルする場合はページ下部の [Cancel (キャンセル)] を選択します。
7. [Waiting period (in days)] (待機期間 (日数)) に、日数として 7~30 の値を入力します。
8. 削除する KMS キーを確認します。
9. [Confirm you want to schedule this key for deletion in **<number of days>** days] (このキーを <日数> 日後に削除するスケジュールを確定する) の横にあるチェックボックスをオンにします。
10. [Schedule deletion] (削除をスケジュールする) を選択します。

KMS キーのステータスが [Pending deletion] (削除保留中) に変わります。

AWS KMS API を使用する場合

次の例のように、 [aws kms schedule-key-deletion](#) コマンドを使用して、 [カスターマネージドキー](#) からキーの削除をスケジュールします。

AWS マネージドキー または の削除をスケジュールすることはできません AWS 所有のキー。

```
$ aws kms schedule-key-deletion --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --  
pending-window-in-days 10
```

正常に使用すると、 は次の例に示す出力のような出力 AWS CLI を返します。

```
{  
  "KeyId": "arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
  "DeletionDate": 1598304792.0,  
  "KeyState": "PendingDeletion",  
  "PendingWindowInDays": 10  
}
```

キー削除をキャンセルする

[KMS キーの削除をスケジュール](#)した後、キー削除がまだ[削除待ち](#)の状態であればキー削除をキャンセルすることができます。キー削除は、AWS KMS コンソールでキャンセルするか、もしくは [CancelKeyDeletion](#) オペレーションを使用してキャンセルできます。削除待ち状態の KMS キーの削除をキャンセルすると、その KMS キーのキー状態は Disabled になります。KMS キーの有効化の詳細については、「[キーの有効化と無効化](#)」を参照してください。

AWS KMS コンソールの使用

キーの削除をキャンセルするには

1. <https://console.aws.amazon.com/kms> で AWS KMS コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. 復元する KMS キーの横にあるチェックボックスをオンにします。
5. [Key actions] (キーのアクション)、[Cancel key deletion] (キーの削除をキャンセル) の順に選択します。

KMS キーのステータスが [Pending deletion] (削除保留中) から [Disabled] (無効) に変わります。KMS キーを使用するには、[有効化する](#)必要があります。

AWS KMS API の使用

次の例 AWS CLI に示すように、[aws kms cancel-key-deletion](#) コマンドを使用して からのキーの削除をキャンセルします。

```
$ aws kms cancel-key-deletion --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

正常に使用すると、 は次の例に示す出力のような出力 AWS CLI を返します。

```
{
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

KMS キーのステータスが [Pending Deletion] (削除保留中) から [Disabled] (無効) に変わります。KMS キーを使用するには、[有効化する](#)必要があります。

削除待ち状態の KMS キーの使用を検出するアラームの作成

AWS CloudTrail、Amazon CloudWatch Logs、Amazon Simple Notification Service (Amazon SNS) の機能を組み合わせて、アカウント内の誰かが削除保留中の KMS キーを使用しようとしたときに通知する Amazon CloudWatch アラームを作成できます。この通知を受け取った場合は、KMS キーの削除をキャンセルして、削除する決定を検討し直す必要があります。

以下の手順では、「**Key ARN** is pending deletion」エラーメッセージが CloudTrail ログファイルに書き込まれるたびに通知を受信するアラームの作成方法について説明します。このエラーメッセージは、[暗号化オペレーション](#)で、ユーザーまたはアプリケーションが KMS キーの使用を試みたことを示します。この通知はエラーメッセージにリンクされているため、ListKeys、CancelKeyDeletion、PutKeyPolicy などの削除保留中の KMS キーで許可される API オペレーションを使用するときにはトリガーされません。このエラーメッセージを返す AWS KMS API オペレーションのリストを確認するには、「」を参照してください[キーの AWS KMS キーステータス](#)。

受信した E メール通知には、KMS キーや暗号化オペレーションは表示されません。この情報は、[CloudTrail ログ](#)で見つけることができます。その代わりに、アラームの状態が [OK] から [アラーム] に変わったことが E メールで報告されます。CloudWatch アラームと、状態の変更の詳細については、「Amazon CloudWatch ユーザーガイド」の「[Amazon CloudWatch アラームの使用](#)」を参照してください。

⚠ Warning

この Amazon CloudWatch アラームは、AWS KMS の外部にある非対称 KMS キーの公開キーの使用は検出できません。公開キーの暗号化に使用される非対称 KMS キーを削除する際の特別なリスク (復号できない暗号テキストの作成など) の詳細については、[Deleting asymmetric KMS keys](#) を参照してください。

この手順では、削除保留中のインスタンスを検索する CloudWatch ロググループのメトリクススフィアを作成します。次に、ロググループのメトリクスに基づいて CloudWatch アラームを作成します。詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「[フィルターを使用したログイベントからのメトリクスの作成](#)」を参照してください。

1. CloudTrail ログを解析する CloudWatch メトリクススフィアを作成します。

以下の必須値を使用して、「[ロググループのメトリクススフィアを作成する](#)」の手順に従ってください。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
フィルターパターン	{ \$.eventSource = kms* && \$.errorMessage = "* is pending deletion." }
メトリクス値	1

2. ステップ 1 で作成したメトリクススフィアに基づいて CloudWatch アラームを作成します。

「次の必須値を使用して[ロググループメトリクススフィアに基づいて CloudWatch アラームを作成する](#)」の手順に従います。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
メトリクスフィルター	ステップ 1 で作成したメトリクススフィアの名前。

フィールド	値
しきい値タイプ	静的
条件	<i>metric-name</i> が より大きい/等しい場合 1
アラームのデータポイント	1 のうち 1
欠損データ処理	欠損データを良好 (しきい値に違反していない) として扱う

この手順を完了すると、新しい CloudWatch アラームが ALARM 状態に入るたびに通知が届きます。このアラームの通知を受信した場合は、データの暗号化または復号化に削除が予定されている KMS キーがまだ必要であることを意味します。その場合は、[KMS キーの削除をキャンセルし](#)、削除する決定を検討し直す必要があります。

KMS キーの過去の使用状況を確認する

KMS キーを削除する前に、そのキーで暗号化された暗号文の数を知りたい場合があります。AWS KMS はこの情報を保存せず、暗号文も保存しません。KMS キーの過去の使用状況を把握することで、後で必要になるかどうかを判断できる場合があります。このトピックでは、KMS キーの過去の使用状況の確認に役立つ複数の戦略を示します。

Warning

過去と実際の使用状況を判断するためのこれらの戦略は、AWS ユーザーと AWS KMS オペレーションにのみ有効です。AWS KMS の外部で非対称 KMS キーの公開キーの使用を検出することはできません。公開キーの暗号化に使用される非対称 KMS キーを削除する際の特別なリスク (復号できない暗号テキストの作成など) の詳細については、[Deleting asymmetric KMS keys](#) を参照してください。

トピック

- [KMS キーのアクセス許可内容を確認し、潜在的な使用の範囲を判断する](#)

- [AWS CloudTrail ログを調べて実際の使用状況を判断する](#)

KMS キーのアクセス許可内容を確認し、潜在的な使用の範囲を判断する

現在、KMS キーにアクセスできるユーザーやアプリケーションを明らかにすることで、KMS キーが使用されている範囲や、今後必要かどうかを判断できる場合があります。現在、KMS キーにアクセスできるユーザーやアプリケーションを明らかにする方法については、[へのアクセスの確認 AWS KMS keys](#) を参照してください。

AWS CloudTrail ログを調べて実際の使用状況を判断する

KMS キーの使用履歴を使用して、特定の KMS キーで暗号化された暗号化テキストがあるかどうかを判断できます。

すべての AWS KMS API アクティビティは AWS CloudTrail ログファイルに記録されます。KMS キーがあるリージョンで [CloudTrail 証跡を作成](#) した場合は、CloudTrail ログファイルを調べて、特定の KMS キーのすべての AWS KMS API アクティビティの履歴を表示できます。トレイルを持っていない場合でも、[CloudTrail イベント履歴で最近のイベント](#) を表示できます。が CloudTrail AWS KMS を使用方法の詳細については、「」を参照してください [を使用した AWS KMS API コールのログ記録 AWS CloudTrail](#)。

次の例は、KMS キーを使用して、Amazon Simple Storage Service (Amazon S3) に保存されているオブジェクトを保護するときに生成される CloudTrail ログエントリを示しています。この例では、[KMS キー \(SSE-KMS\) を使用したサーバー側の暗号化を使用してデータを保護し](#)、オブジェクトを Amazon S3 にアップロードします。SSE-KMS を使用して Amazon S3 にオブジェクトをアップロードする場合は、オブジェクトの保護に使用する KMS キーを指定します。Amazon S3 は AWS KMS [GenerateDataKey](#) オペレーションを使用してオブジェクトの一意的データキーをリクエストし、このリクエストイベントは次のようなエントリで CloudTrail に記録されます。

```
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROACKCEVSQ6C2EXAMPLE:example-user",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admins/example-user",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
```

```

    "creationDate": "2015-09-10T23:12:48Z"
  },
  "sessionIssuer": {
    "type": "Role",
    "principalId": "AROACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:role/Admins",
    "accountId": "111122223333",
    "userName": "Admins"
  }
},
"invokedBy": "internal.amazonaws.com"
},
"eventTime": "2015-09-10T23:58:18Z",
"eventSource": "kms.amazonaws.com",
"eventName": "GenerateDataKey",
"awsRegion": "us-west-2",
"sourceIPAddress": "internal.amazonaws.com",
"userAgent": "internal.amazonaws.com",
"requestParameters": {
  "encryptionContext": {"aws:s3:arn": "arn:aws:s3:::example_bucket/example_object"},
  "keySpec": "AES_256",
  "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
},
"responseElements": null,
"requestID": "cea04450-5817-11e5-85aa-97ce46071236",
"eventID": "80721262-21a5-49b9-8b63-28740e7ce9c9",
"readOnly": true,
"resources": [{
  "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "accountId": "111122223333"
}],
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

後で Amazon S3 からこのオブジェクトをダウンロードすると、Amazon S3 は指定された KMS キーを使用してオブジェクトのデータキーを復号 AWS KMS する Decrypt リクエストを に送信します。これを行うと、CloudTrail ログファイルに次のようなエントリが含まれます。

```

{
  "eventVersion": "1.02",

```

```

"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AROACKCEVSQ6C2EXAMPLE:example-user",
  "arn": "arn:aws:sts::111122223333:assumed-role/Admins/example-user",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2015-09-10T23:12:48Z"
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AROACKCEVSQ6C2EXAMPLE",
      "arn": "arn:aws:iam::111122223333:role/Admins",
      "accountId": "111122223333",
      "userName": "Admins"
    }
  },
  "invokedBy": "internal.amazonaws.com"
},
"eventTime": "2015-09-10T23:58:39Z",
"eventSource": "kms.amazonaws.com",
"eventName": "Decrypt",
"awsRegion": "us-west-2",
"sourceIPAddress": "internal.amazonaws.com",
"userAgent": "internal.amazonaws.com",
"requestParameters": {
  "encryptionContext": {"aws:s3:arn": "arn:aws:s3:::example_bucket/example_object"}},
"responseElements": null,
"requestID": "db750745-5817-11e5-93a6-5b87e27d91a0",
"eventID": "ae551b19-8a09-4cfc-a249-205ddba330e3",
"readOnly": true,
"resources": [{
  "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "accountId": "111122223333"
}],
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

すべての AWS KMS API アクティビティは CloudTrail によってログに記録されます。これらのログ エントリを評価すると、特定の KMS キーの過去の使用状況を明らかにし、その KMS キーを削除する必要があるかどうかを判断できる場合があります。

CloudTrail ログファイルに AWS KMS API アクティビティがどのように表示されるかの例については、「」を参照してください[を使用した AWS KMS API コールのログ記録 AWS CloudTrail](#)。CloudTrail の詳細については、[AWS CloudTrail ユーザーガイド](#)を参照してください。

インポートされたキーマテリアルを削除する

インポートされたキーマテリアルは、KMS キーからいつでも削除できます。また、インポートされたキーマテリアルの有効期限が切れると、 はキーマテリアル AWS KMS を削除します。いずれの場合も、キーマテリアルが削除されると、KMS キーの [キー状態](#) がインポート保留中になり、KMS キーを暗号化オペレーションで使用することはできません。

単一リージョンの対称暗号化キーには、複数のキーマテリアルを関連付けて、キーの状態をインポート保留中 PENDING_ROTATION に変更する以外の状態のキーマテリアルを削除または期限切れにすることができます。これらのキーの場合、KMS は各キーマテリアルに一意の識別子を割り当てます。[ListKeyRotations](#) API を使用して、これらのキーマテリアル識別子を表示できます。[DeleteImportedKeyMaterial](#) API の key-material-id パラメータを使用して識別子を指定することで、特定のキーマテリアルを削除できます。

Warning

key-material-id パラメータはオプションであり、指定しない場合、AWS KMS は現在のキーマテリアルを削除します。

KMS キーの無効化とアクセス許可の取り消しに加えて、キーマテリアルの削除は、KMS キーの使用をすぐに、しかし一時的に停止する戦略として使用できます。これとは対照的に、キーマテリアルがインポートされた KMS キーの削除をスケジュールすることでも、KMS キーの使用をすぐに停止できます。ただし、待機期間中に削除がキャンセルされない場合、KMS キー、関連するキーマテリアル、およびすべてのキーマテリアルデータは完全に削除されます。詳細については、「[Deleting KMS keys with imported key material](#)」を参照してください。

キーマテリアルを削除するには、AWS KMS コンソールまたは [DeleteImportedKeyMaterial](#) API オペレーションを使用できます。 は、 [インポートされたキーマテリアルを削除する](#) とき、および [AWS KMS 期限切れのキーマテリアルを削除する](#) ときに AWS CloudTrail、ログにエントリ AWS KMS を記録します。

キーマテリアルの削除が AWS サービスに与える影響

キーマテリアルを削除すると、KMS キーはすぐに使用できなくなります (結果整合性があります)。ただし、KMS キーで保護された[データキー](#)により暗号化されたリソースは、KMS キーがデータキーの復号化などで再び使用されるまでは影響を受けません。この問題は AWS のサービスに影響します。その多くが、リソースを保護するためにデータキーを使用しています。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

AWS KMS コンソールの使用

AWS KMS コンソールを使用して、キーマテリアルを削除できます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。
4. 次のいずれかを行います：
 - キーマテリアルがインポートされた KMS キーのチェックボックスをオンにします。[キーのアクション]、[キーマテリアルの削除] を選択します。複数のキーマテリアルが関連付けられている対称暗号化キーの場合、現在のキーマテリアルは削除されます。
 - インポートされたキーマテリアルを持つ単一リージョンの対称暗号化 KMS キーの場合は、KMS キーのエイリアスまたはキー ID を選択します。キーマテリアルとローテーションタブを選択します。キーマテリアルテーブルには、キーに関連付けられたすべてのキーマテリアルが一覧表示されます。削除するキーマテリアルに対応する行のアクションメニューからキーマテリアルの削除を選択します。
5. キーマテリアルを削除することを確認してから、[キーマテリアルの削除] を選択します。KMS キーのステータスに対応するその[キーステータス](#)は、インポート保留中に変わります。削除されたキーマテリアルが PENDING_ROTATION 状態であった場合、KMS キーのステータスに変更はありません。

AWS KMS API の使用

[AWS KMS API](#) を使用して、キーマテリアルを削除するには、[DeleteImportedKeyMaterial](#) リクエストを送信します。次の例では、[AWS CLI](#) を使用してこのオペレーションを行う方法を示します。

1234abcd-12ab-34cd-56ef-1234567890ab を、削除する予定のキーマテリアルを持つ KMS キーのキー ID と置き換えます。KMS キーのキー ID または ARN を使用できますが、このオペレーションにエイリアスを使用することはできません。次のコマンドは、キーに関連付けられた唯一のキーマテリアルである可能性がある現在のキーマテリアルを削除します。

```
$ aws kms delete-imported-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

特定のキーマテリアルを削除するには、key-material-id パラメータを使用して識別されたキーマテリアルを指定します。を削除するキーマテリアルの識別子 **123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0** に置き換えます。

```
$ aws kms delete-imported-key-material --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
--key-material-id 123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0
```

データキーを生成する

データキーは、大量のデータ、および他のデータ暗号化キーなどのデータを暗号化するために使用できる暗号化キーです。ダウンロードできない対称 KMS キーとは異なり、データキーは AWS KMS の外部での使用のためにユーザーに返されます。

がデータキー AWS KMS を生成すると、すぐに使用できるプレーンテキストのデータキー (オプション) と、データと共に安全に保存できる暗号化されたデータキーのコピーが返されます。データを復号する準備ができたなら、まず暗号化されたデータキーを復号 AWS KMS するように に依頼します。

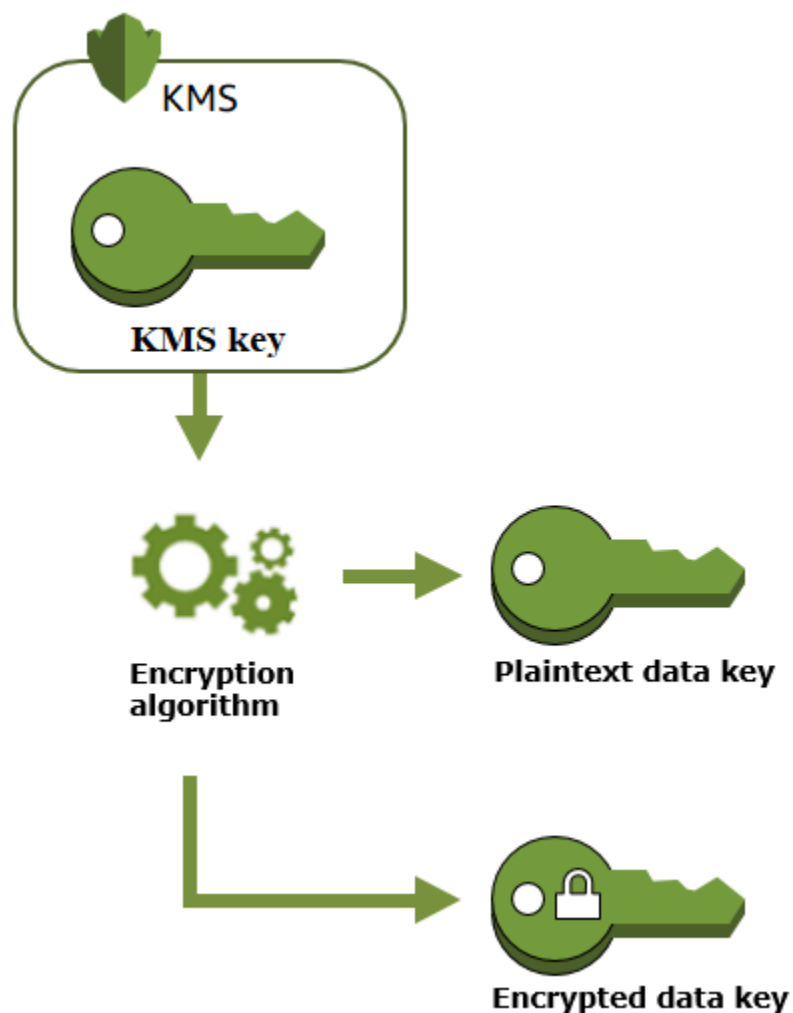
AWS KMS はデータキーを生成、暗号化、復号します。ただし、AWS KMS はデータキーを保存、管理、追跡したり、データキーを使用して暗号化オペレーションを実行したりしません。外部でデータキーを使用および管理する必要があります AWS KMS。データキーを安全に使用方法については、「[AWS Encryption SDK](#)」を参照してください。

トピック

- [データキーの作成](#)
- [データキーを使用した暗号化オペレーションの仕組み](#)
- [使用できない KMS キーがデータキーに及ぼす影響](#)

データキーの作成

データキーを作成するには、[GenerateDataKey](#) オペレーションを呼び出します。データキー AWS KMS を生成します。次に、ユーザーが指定する[対称暗号化 KMS キー](#)のデータキーのコピーを暗号化します。このオペレーションでは、データキーのプレーンテキストコピーと KMS キーで暗号化されたデータキーのコピーが返されます。以下の図では、このオペレーションを示しています。



AWS KMS は、暗号化されたデータキーのみを返す [GenerateDataKeyWithoutPlaintext](#) オペレーションもサポートしています。データキーを使用する必要がある場合は、AWS KMS に[復号](#)を依頼してください。

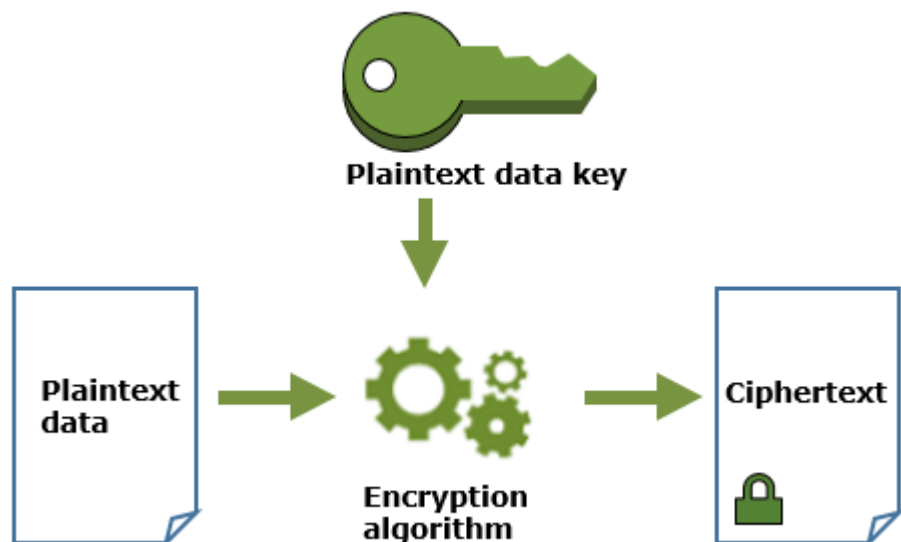
データキーを使用した暗号化オペレーションの仕組み

以下のトピックでは、[GenerateDataKey](#) または [GenerateDataKeyWithoutPlaintext](#) オペレーションによって生成されるデータキーがどのように機能するかについて説明します。

データキーでデータを暗号化する

AWS KMS は、データキーを使用してデータを暗号化することはできません。ただし AWS KMS、OpenSSL や などの暗号化ライブラリを使用するなど、 の外部でデータキーを使用できます [AWS Encryption SDK](#)。

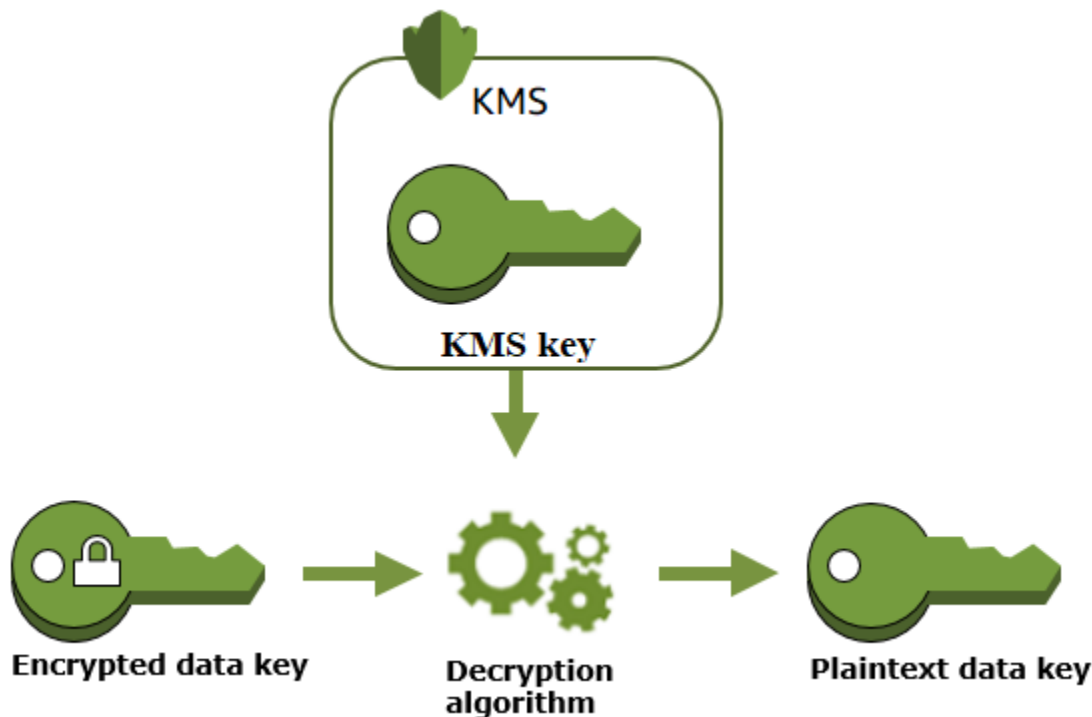
プレーンテキストのデータキーを使用してデータを暗号化したら、できる限り早急にメモリからそれを削除します。暗号化したデータと一緒に暗号化データキーを安全に保存して、データの復号に利用することができます。



データキーでデータを復号する

データを復号するには、暗号化されたデータキーを [Decrypt](#) オペレーションに渡します。AWS KMS は KMS キーを使用してデータキーを復号し、プレーンテキストのデータキーを返します。プレーンテキストのデータキーを使ってデータを復号し、続いてできる限り早急にメモリからプレーンテキストのデータキーを削除します。

以下の図では、Decrypt オペレーションを使用して暗号化されたデータキーを復号する方法を示しています。



使用できない KMS キーがデータキーに及ぼす影響

KMS キーが使用できなくなると、その影響はほぼ即時に表れます (最終的な一貫性の対象となります)。KMS キーの[キーステータス](#)は新しい条件を反映して変化し、[暗号化オペレーション](#)で KMS キーを使用するすべてのリクエストは失敗します。

ただし、KMS キーで暗号化されたデータキー、およびデータキーで暗号化されたデータへの影響は、データキーの復号などに KMS キーが再度使用されるまで遅延します。

KMS キーは、実行する可能性のある次のアクションなどのような、さまざまな原因によって使用できなくなる可能性があります。

- [KMS キーを無効にする](#)
- [KMS キーの削除をスケジュールする](#)
- キーマテリアルがインポートされた KMS キーから[キーマテリアルを削除する](#)か、インポートされたキーマテリアルの有効期限が切れるようにします。EXTERNAL オリジンを持つ KMS キーに複数のキーマテリアルが関連付けられている場合、キーマテリアルを削除または期限切れにすると、キーは使用できなくなります。
- KMS [AWS CloudHSM キーをホストするキーストアを切断](#)するか、KMS [キーのキーマテリアルとして機能する AWS CloudHSM クラスターからキーを削除](#)します。

- KMS キーをホストする[外部キーストアの切断](#)、または外部キーストアプロキシへの暗号化および復号リクエストを妨げるその他のアクション (外部キーマネージャーからの外部キーの削除など) です。

この効果は、サービスが管理するリソースを保護するためにデータキーを使用する多くの AWS のサービス にとって特に重要です。次の例では、Amazon Elastic Block Store (Amazon EBS) と Amazon Elastic Compute Cloud (Amazon EC2) を使用しています。異なる は、異なる方法でデータキー AWS のサービス を使用します。詳細については、AWS のサービスの「セキュリティ」の章の「データ保護」セクションを参照してください。

例えば、次のシナリオが考えられます。

1. [暗号化された EBS ボリュームを作成し](#)、KMS キーを指定して保護します。Amazon EBS は、KMS キーを使用してボリュームの[暗号化されたデータキーを生成する](#)よう、AWS KMS に要求します。Amazon EBS は、暗号化されたデータキーをボリュームのメタデータとともに保存します。
2. EBS ボリュームを EC2 インスタンスにアタッチすると、Amazon EC2 は、KMS キーを使用して EBS ボリュームの暗号化されたデータキーを復号します。Amazon EC2 は、EBS ボリュームに対するすべてのディスク I/O を暗号化する責任を担う Nitro ハードウェア内のデータキーを使用します。データキーは、EBS ボリュームが EC2 インスタンスにアタッチされる限り、Nitro ハードウェア内で維持されます。
3. KMS キーを使用不可能にするアクションを実行しました。これによって EC2 インスタンスまたは EBS ボリュームにただちに影響が出ることはありません。Amazon EC2 は、KMS キーではなくデータキーを使用して、ボリュームがインスタンスにアタッチされている限り、すべてのディスク I/O を暗号化します。
4. ただし、暗号化された EBS ボリュームが EC2 インスタンスからデタッチされると、Amazon EBS は Nitro ハードウェアからデータキーを削除します。次回、暗号化された EBS ボリュームが EC2 インスタンスにアタッチされると、アタッチメントは失敗します。これは、Amazon EBS は KMS キーを使用してボリュームの暗号化されたデータキーを復号できないためです。EBS ボリュームを再度使用するには、KMS キーを再度使用可能にする必要があります。

データキーペアを生成する

非対称 KMS キーは、データキーペアを表します。データキーペアは、数学的に関連するパブリックキーとプライベートキーで構成される非対称データキーです。クライアント側の暗号化と復号、外部の署名と検証 AWS KMS、または 2 つのピア間の共有シークレットの確立に使用するように設計されています。

OpenSSL などのツールが生成するデータキーペアとは異なり、各データキーペアのプライベートキーを、指定したの対称暗号化 KMS キーで AWS KMS 保護 AWS KMS します。ただし、AWS KMS はデータキーペアを保存、管理、追跡したり、データキーペアを使用して暗号化オペレーションを実行したりしません。AWS KMSの外部でデータキーペアを使用して管理する必要があります。

トピック

- [データキーペアを作成する](#)
- [データキーペアを使用した暗号化オペレーションの仕組み](#)

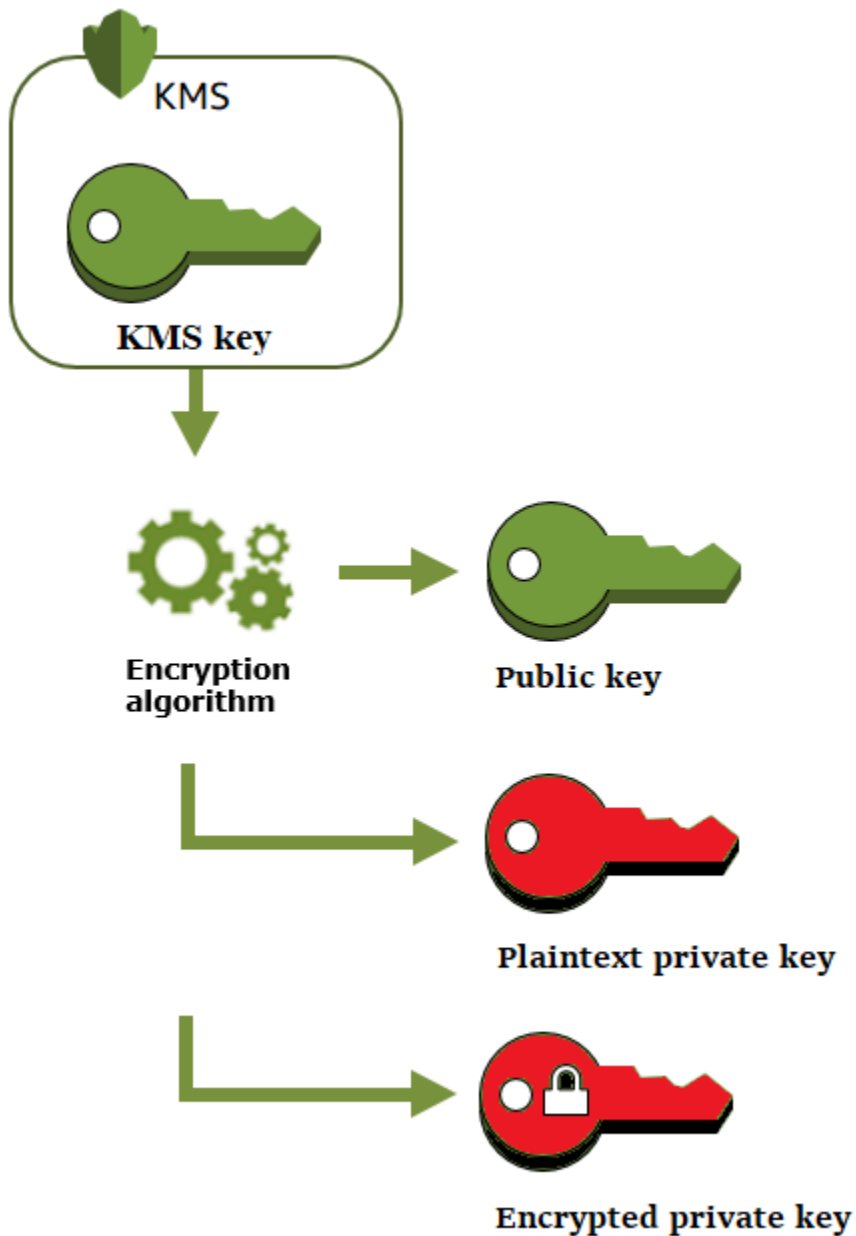
データキーペアを作成する

データキーペアを作成するには、[GenerateDataKeyPair](#) または [GenerateDataKeyPairWithoutPlaintext](#) オペレーションを呼び出します。プライベートキーの暗号化に使用する[対称暗号化 KMS キー](#)を指定します。

`GenerateDataKeyPair` は、プレーンテキストパブリックキー、プレーンテキストプライベートキー、暗号化されたプライベートキーを返します。このオペレーションは、デジタル署名を生成する場合など、プレーンテキストプライベートキーをすぐに必要とする場合に使用します。

`GenerateDataKeyPairWithoutPlaintext` は、プレーンテキストのパブリックキーと暗号化されたプライベートキーを返しますが、プレーンテキストプライベートキーは返しません。このオペレーションは、パブリックキーで暗号化する場合など、プレーンテキストプライベートキーがすぐに必要ない場合に使用します。後で、データを復号化するために平文の秘密鍵が必要な場合は、[Decrypt](#) オペレーションを呼び出すことができます。

以下の図では、`GenerateDataKeyPair` オペレーションを示しています。`GenerateDataKeyPairWithoutPlaintext` オペレーションは、プレーンテキストプライベートキーを省略します。



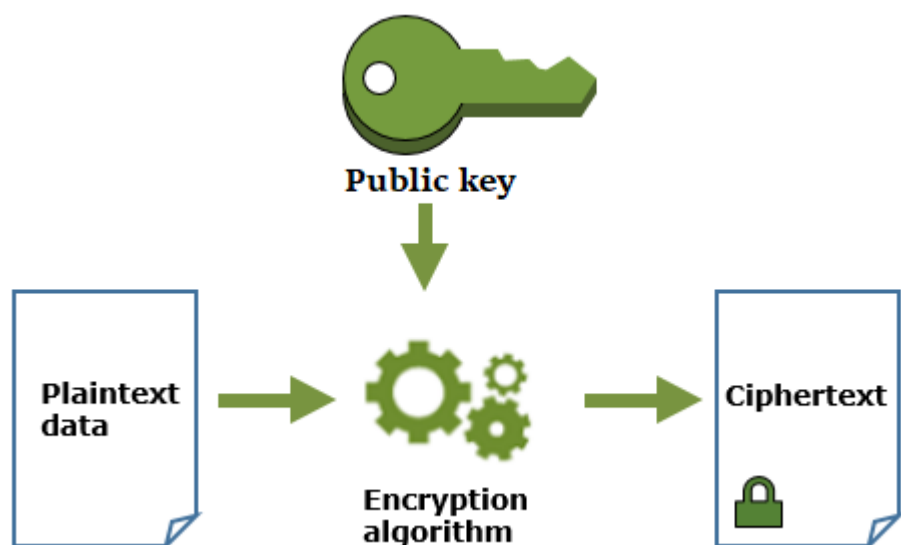
データキーペアを使用した暗号化オペレーションの仕組み

以下のトピックでは、[GenerateDataKeyPair](#) または [GenerateDataKeyPairWithoutPlaintext](#) オペレーションによって生成されたデータキーペアで実行できる暗号化オペレーションとその仕組みについて説明します。

データキーペアでデータを暗号化する

データキーペアを使用して暗号化する場合、ペアのパブリックキーを使用してデータを暗号化し、同じペアのプライベートキーを使用してデータを復号します。通常、データキーペアは、プライベートキーを保持している当事者のみが復号できるデータを、多くの当事者が暗号化する必要がある場合に使用されます。

次の図に示すように、パブリックキーを持つ当事者は、そのキーを使用してデータを暗号化します。

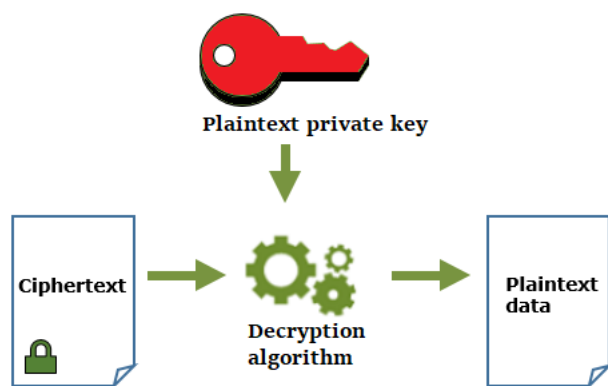


データキーペアでデータを復号する

データを復号するには、データキーペアでプライベートキーを使用します。オペレーションを成功させるには、パブリックキーとプライベートキーが同じデータキーペアのものである必要があり、また、同じ暗号化アルゴリズムを使用する必要があります。

暗号化された秘密鍵を復号化するには、[復号](#)オペレーションに渡します。プレーンテキストプライベートキーを使用してデータを復号します。その後、できるだけ早くプレーンテキストのプライベートキーをメモリから削除します。

次の図は、データキーペアのプライベートキーを使用して暗号テキストを復号する方法を示しています。



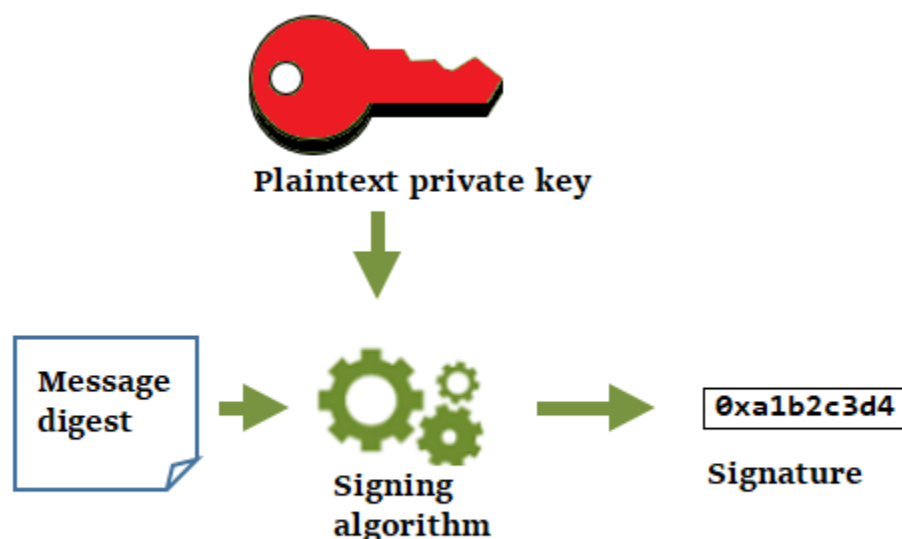
データキーペアでメッセージに署名する

メッセージの暗号化署名を生成するには、データキーペアのプライベートキーを使用します。パブリックキーを持つすべてのユーザーは、メッセージが自分のプライベートキーで署名されたこと、および署名されてから変更されていないことを確認するために使用できます。

プライベートキーを暗号化する場合は、暗号化されたプライベートキーを [Decrypt](#) オペレーションに渡します。AWS KMS は KMS キーを使用してデータキーを復号し、プレーンテキストのプライベートキーを返します。署名を生成するには、プレーンテキストプライベートキーを使用します。その後、できるだけ早くプレーンテキストのプライベートキーをメモリから削除します。

メッセージに署名するには、OpenSSL で [dgst](#) コマンドなどの暗号化ハッシュ関数を使用してメッセージダイジェストを作成します。次に、プレーンテキストプライベートキーを署名アルゴリズムに渡します。結果は、メッセージの内容を表す署名です。(最初にダイジェストを作成しなくても、短いメッセージで署名できる場合があります。メッセージの最大サイズは、使用する署名ツールによって異なります)。

次の図は、データキーペアのプライベートキーを使用してメッセージに署名する方法を示しています。

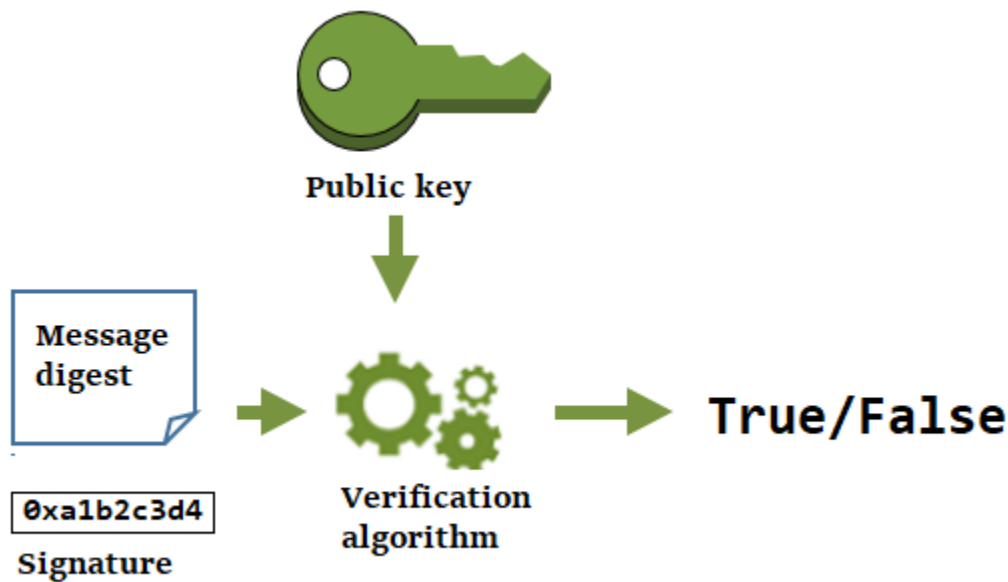


データキーペアを使用した署名の検証

データキーペアにパブリックキーを持っているユーザーは、誰でもそれを使用して、プライベートキーで生成した署名を検証できます。検証では、承認されたユーザーが指定したプライベートキーと署名アルゴリズムを使用してメッセージに署名し、メッセージが署名後に変更されていないことを確認します。

署名を検証する当事者は、同じタイプのダイジェストを生成し、同じアルゴリズムを使用して、メッセージの署名に使用されるプライベートキーに対応するパブリックキーを使用する必要があります。

次の図は、データキーペアのパブリックキーを使用してメッセージ署名を検証する方法を示しています。



データキーペアを用いて共有シークレットを取得する

キーアグリーメントにより、それぞれ楕円曲線のパブリック/プライベートキーペアを持つ 2 つのピアが、安全でないチャネル上で共有シークレットを確立することができます。[共有シークレットを取得する](#)には、2 つのピアが、安全でない通信チャネル (インターネットなど) を介してパブリックキーを交換する必要があります。次に、各パーティがそれぞれのプライベートキーと自ピアのパブリックキーを使用することで、キーアグリーメントアルゴリズムを使用して同じ共有シークレットを計算します。この共有シークレット値を使用して、2 つのピア間で送信されるデータを暗号化および復号、または HMAC を生成および検証できる対称キーを取得できます。

Note

AWS KMS では、パブリックキーを使用して共有シークレットを取得する前に、受信したパブリックキーが想定された当事者からのものであることを確認することを強くお勧めします。

パブリックキーを使用してオフラインオペレーションを実行する

非対称 KMS キーでは、プライベートキーは作成 AWS KMS され、暗号化されていないままになる AWS KMS ことはありません。プライベートキーを使用するには、呼び出す必要があります AWS KMS。API AWS KMS オペレーションを呼び出す AWS KMS ことで、内でパブリックキーを使用できます。または、[パブリックキーをダウンロード](#)して、の外部で使用するために共有することもできます AWS KMS。

パブリックキーを共有して、プライベートキーでのみ復号 AWS KMS できる 以外のデータを他のユーザーに暗号化させることができます。または、プライベートキーを使用して生成した AWS KMS の外部にあるデジタル署名を他のユーザーが確認できるようにすることができます。または、パブリックキーをピアと共有して共有シークレットを取得することもできます。

AWS KMS内の非対称 KMS キーでパブリックキーを使用すると、すべての AWS KMS オペレーションの一部である認証、認可、ロギングの利点を得ることができます。また、復号できないデータを暗号化するリスクも軽減します。これらの機能は、の外部では有効ではありません AWS KMS。詳細については、「[パブリックキーのダウンロードに関する特別な考慮事項](#)」を参照してください。

Tip

データキーまたは SSH キーをお探しですか。このトピックでは、プライベートキーをエクスポートすることができない AWS Key Management Serviceでの非対称キーの管理方法を説明しています。プライベートキーが対称暗号化 KMS キーで保護されているエクスポート可能なデータキーペアについては、「[GenerateDataKeyPair](#)」を参照してください。Amazon EC2 インスタンスに関連付けられたパブリックキーのダウンロード方法については、「[Amazon EC2 ユーザーガイド](#)」内の「パブリックキーの取得」および「[Amazon EC2 ユーザーガイド](#)」を参照してください。

トピック

- [パブリックキーのダウンロードに関する特別な考慮事項](#)
- [パブリックキーをダウンロードする](#)
- [オフラインオペレーションの例](#)

パブリックキーのダウンロードに関する特別な考慮事項

KMS キーを保護するために、はすべてのオペレーションのアクセスコントロール、認証された暗号化、および詳細なログ AWS KMS を提供します。AWS KMS また、では、KMS キーの使用を一時的または永続的に防止することもできます。最後に、AWS KMS オペレーションは、復号できないデータを暗号化するリスクを最小限に抑えるように設計されています。これらの機能は、ダウンロードしたパブリックキーを の外部で使用する場合は使用できません AWS KMS。

Authorization

内の [KMS キーへのアクセスを制御するキーポリシー](#)と [IAM ポリシー](#) AWS KMS は、 の外部で実行されるオペレーションには影響しません AWS。パブリックキーを取得できるユーザーは、KMS キーを使用してデータを暗号化したり、署名を検証したりするアクセス許可がない場合 AWS KMS でも、 の外部で使用できます。

キーの用途の制限

キーの使用制限は、 の外部では有効ではありません AWS KMS。KeyUsage の を持つ KMS キーを使用して [Encrypt](#) オペレーションを呼び出すとSIGN_VERIFY、AWS KMS オペレーションは失敗します。ただし、SIGN_VERIFYまたは KeyUsageの を持つ KMS キーからパブリックキー AWS KMS を使用して の外部でデータを暗号化する場合KEY_AGREEMENT、データを復号することはできません。

アルゴリズムの制限

AWS KMS がサポートする暗号化アルゴリズムと署名アルゴリズムの制限は、 の外部では有効ではありません AWS KMS。外部で KMS キーからパブリックキーを使用してデータを暗号化し AWS KMS、AWS KMS がサポートしていない暗号化アルゴリズムを使用する場合、データを復号することはできません。

KMS キーの無効化と削除

内の暗号化オペレーションで KMS キーが使用されないようにするために実行できるアクション AWS KMS は、誰も の外部でパブリックキーを使用することを妨げません AWS KMS。例えば、KMS キーの無効化、KMS キーの削除のスケジューリング、KMS キーの削除、KMS キーからのキーマテリアルの削除は、AWS KMSの外部のパブリックキーには影響しません。非対称 KMS キーを削除した場合、またはそのキーマテリアルを削除または紛失した場合、 の外部にあるパブリックキーで暗号化したデータは AWS KMS 回復できません。

ログ記録

AWS CloudTrail リクエスト、レスポンス、日付、時刻、承認されたユーザーなど、すべての AWS KMS オペレーションを記録する ログは、 の外部でのパブリックキーの使用を記録しません AWS KMS。

SM2 キーペアによるオフライン検証 (中国リージョンのみ)

SM2 パブリックキー AWS KMS を使用して の外部の署名を検証するには、識別 ID を指定する必要があります。デフォルトでは、AWS KMS を識別 ID 1234567812345678として使用します。詳細については、「[SM2 キーペアによるオフライン検証](#)」(中国リージョンのみ) を参照してください。

パブリックキーをダウンロードする

コンソールの非対称 KMS キーペアから、AWS KMS または [GetPublicKey](#) オペレーションを使用して、パブリックキーをダウンロードできます。パブリックキーをダウンロードするには、非対称 KMS キーに対する `kms:GetPublicKey` アクセス許可が必要です。

が AWS KMS 返すパブリックキーは、RFC 5280 で定義されているように、SubjectPublicKeyInfo (SPKI) と呼ばれる DER エンコードされた X.509 パブリックキーです。 <https://datatracker.ietf.org/doc/html/rfc5280> HTTP API または を使用する場合 AWS CLI、値は Base64-encodedされます。それ以外の場合は、値は Base64 エンコードされていません。

非対称 KMS キーペアからパブリックキーをダウンロードするには、`kms:GetPublicKey` アクセス許可が必要です。アクセス AWS KMS 許可の詳細については、「」を参照してください [アクセス許可に関するリファレンス](#)。

AWS KMS コンソールの使用

を使用して、 の非対称 KMS キーからパブリックキー AWS Management Console を表示、コピー、ダウンロードできます AWS アカウント。異なる の非対称 KMS キーからパブリックキーをダウンロードするには AWS アカウント、AWS KMS API を使用します。

1. にサインイン AWS Management Console し、 <https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。
4. 非対称 KMS キーのエイリアスまたはキー ID を選択します。

5. [Cryptographic configuration] (暗号化の設定) タブを選択します。[Key spec] (キー仕様)、[Key usage] (キーの用途)、[Encryption algorithms] (暗号化アルゴリズム)、または [Signing Algorithms] (署名アルゴリズム) フィールドの値を記録します。外部でパブリックキーを使用するには、これらの値を使用する必要があります AWS KMS。パブリックキーを共有するときは、必ずこの情報を共有してください。
6. [Public key] (パブリックキー) タブを選択します。
7. パブリックキーをクリップボードにコピーするには、[Copy] (コピー) を選択します。パブリックキーをファイルにダウンロードするには、[Download] (ダウンロード) を選択します。

AWS KMS API の使用

[GetPublicKey](#) オペレーションは、非対称 KMS キーでパブリックキーを返します。また、キーの使用法や暗号化アルゴリズムなど AWS KMS、の外部でパブリックキーを正しく使用するために必要な重要な情報も返します。これらの値を保存し、パブリックキーを共有する場合は必ず共有してください。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

KMS キーを指定するには、その [キー ID](#)、[キー ARN](#)、[エイリアス名](#)、[エイリアス ARN](#) を使用します。エイリアス名を使用する場合は、接頭辞として `alias/` を付けます。別の で KMS キーを指定するには AWS アカウント、そのキー ARN またはエイリアス ARN を使用する必要があります。

このコマンドを実行する前に、サンプルのエイリアス名を KMS キーの有効な識別子に置き換えます。このコマンドを実行するには、KMS キーに対する `kms:GetPublicKey` アクセス許可が必要です。

```
$ aws kms get-public-key --key-id alias/example_RSA_3072

{
  "KeySpec": "RSA_3072",
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyUsage": "ENCRYPT_DECRYPT",
  "EncryptionAlgorithms": [
    "RSAES_OAEP_SHA_1",
    "RSAES_OAEP_SHA_256"
  ],
  "PublicKey": "MIIBojANBgkqhkiG..."
```

```
}
```

オフラインオペレーションの例

非対称 KMS キーペアの[パブリックキーをダウンロード](#)したら、それを他のユーザーと共有し、オフラインオペレーションの実行に使用することができます。

AWS CloudTrail リクエスト、レスポンス、日付、時刻、承認されたユーザーなど、すべての AWS KMS オペレーションを記録する ログは、 の外部でパブリックキーの使用を記録しません AWS KMS。

このトピックでは、オフラインオペレーションの例と、オフラインオペレーションを簡単にするためにツール AWS KMS が提供する詳細について説明します。

トピック

- [オフラインで共有シークレットを取得する](#)
- [ML-DSA キーペアによるオフライン検証](#)
- [SM2 キーペアによるオフライン検証 \(中国リージョンのみ\)](#)

オフラインで共有シークレットを取得する

ECC キーペアの[パブリックキーをダウンロード](#)し、オフラインオペレーション、つまり AWS KMS の外部でのオペレーションに使用することができます。

次の [OpenSSL](#) チュートリアルでは、ECC KMS キーペアのパブリックキーと OpenSSL で作成されたプライベートキー AWS KMS を使用して、 外で共有シークレットを取得する 1 つの方法を示します。

1. OpenSSL で ECC キーペアを作成し、 で使用する準備をします AWS KMS。

```
// Create an ECC key pair in OpenSSL and save the private key in
openssl_ecc_key_priv.pem
export OPENSSL_CURVE_NAME="P-256"
export KMS_CURVE_NAME="ECC_NIST_P256"

export OPENSSL_KEY1_PRIV_PEM="openssl_ecc_key1_priv.pem"
openssl ecparam -name ${OPENSSL_CURVE_NAME} -genkey -out ${OPENSSL_KEY1_PRIV_PEM}
```

```
// Derive the public key from the private key
export OPENSSL_KEY1_PUB_PEM="openssl_ecc_key1_pub.pem"
openssl ec -in ${OPENSSL_KEY1_PRIV_PEM} -pubout -outform pem \
    -out ${OPENSSL_KEY1_PUB_PEM}

// View the PEM file containing the public key and extract the public key as a
// Base64 encoded string into OPENSSL_KEY1_PUB_BASE64 for use with AWS KMS
export OPENSSL_KEY1_PUB_BASE64=`cat ${OPENSSL_KEY1_PUB_PEM} | \
    tee /dev/stderr | grep -v "PUBLIC KEY" | tr -d "\n"`
```

2. で ECC キーアグリーメントキーペアを作成し AWS KMS、OpenSSL で使用する準備をします。

```
// Create a KMS key on the same curve as the key pair from step 1
// with a key usage of KEY_AGREEMENT
// Save its ARN in KMS_KEY1_ARN.
export KMS_KEY1_ARN=`aws kms create-key --key-spec ${KMS_CURVE_NAME} \
    --key-usage KEY_AGREEMENT | tee /dev/stderr | jq -r .KeyMetadata.Arn`

// Download the public key and save the Base64-encoded version in KMS_KEY1_PUB_BASE64

export KMS_KEY1_PUB_BASE64=`aws kms get-public-key --key-id ${KMS_KEY1_ARN} | \
    tee /dev/stderr | jq -r .PublicKey`

// Create a PEM file for the public KMS key for use with OpenSSL
export KMS_KEY1_PUB_PEM="aws_kms_ecdh_key1_pub.pem"
echo "-----BEGIN PUBLIC KEY-----" > ${KMS_KEY1_PUB_PEM}
echo ${KMS_KEY1_PUB_BASE64} | fold -w 64 >> ${KMS_KEY1_PUB_PEM}
echo "-----END PUBLIC KEY-----" >> ${KMS_KEY1_PUB_PEM}
```

3. OpenSSL のプライベートキーとパブリック KMS キーを使用して、OpenSSL で共有シークレットを取得します。

```
export OPENSSL_SHARED_SECRET1_BIN="openssl_shared_secret1.bin"
openssl pkeyutl -derive -inkey ${OPENSSL_KEY1_PRIV_PEM} \
    -peerkey ${KMS_KEY1_PUB_PEM} -out ${OPENSSL_SHARED_SECRET1_BIN}
```

ML-DSA キーペアによるオフライン検証

AWS KMS は、最大 4 KB バイトのメッセージについて[連邦情報処理規格 \(FIPS\) 204 標準](#)セクション 3.4 で説明されているように、ML-DSA 署名のヘッジされたバリエーションをサポートします。

4 KB を超えるメッセージに署名するには、メッセージの前処理ステップを の外部で実行します AWS KMS。このハッシュステップでは、NIST FIPS 204 セクション 6.2 で定義されているように、64 バイトのメッセージ代表的な μ を作成します。

AWS KMS には、4 KB を超えるメッセージEXTERNAL_MUに対して というメッセージタイプがあります。RAW メッセージタイプの代わりにこれを使用すると、次のようになります AWS KMS。

- ハッシュステップをすでに実行していることを前提としています
- 内部ハッシュプロセスをスキップする
- 任意のサイズのメッセージに対応

メッセージを検証する場合、使用する方法は、外部システムまたはライブラリのサイズ制限と、64 バイトのメッセージ代表 μ をサポートしているかどうかによって異なります。

- メッセージのサイズ制限より小さい場合は、RAWメッセージタイプを使用します。
- メッセージのサイズ制限より大きい場合は、外部システムで代表的な μ を使用します。

以下のセクションでは、 を使用してメッセージに署名 AWS KMS し、OpenSSL を使用してメッセージを検証する方法を示します。によって課される 4 KB のメッセージサイズ制限の下と上の両方のメッセージの例を示します AWS KMS。OpenSSL は、検証のためにメッセージサイズに制限を課しません。

どちらの例でも、まずパブリックキーを取得します AWS KMS。次の AWS CLI コマンドを使用します。

```
aws kms get-public-key \
  --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
  --output text \
  --query PublicKey | base64 --decode > public_key.der
```

メッセージサイズが 4KB 未満

4 KB 未満のメッセージの場合は、 でRAWメッセージタイプを使用します AWS KMS。を使用できますがEXTERNAL_MU、サイズ制限内のメッセージには必要ありません。

次の AWS CLI コマンドを使用してメッセージに署名します。

```
aws kms sign \
```



```
--key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \
--message 'your message' \
--message-type RAW \
--signing-algorithm ML_DSA_SHAKE_256 \
--output text \
--query Signature | base64 --decode > ExampleSignature.bin
```

OpenSSL を使用してこのメッセージを確認するには、次のコマンドを使用します。

```
echo -n 'your message' | ./openssl dgst -verify public_key.der -signature
ExampleSignature.bin
```

4KB を超えるメッセージサイズ

4KB を超えるメッセージに署名するには、EXTERNAL_MUメッセージタイプを使用します。を使用する場合はEXTERNAL_MU、NIST FIPS 204 セクション 6.2 で定義されている 64 バイトの代表的な μ にメッセージを外部で事前にハッシュし、署名または検証オペレーションに渡します。これは、NIST FIPS 204 セクション 5.4 で定義されている「Pre-hash MLDSA」または HashML-DSA とは異なることに注意してください。

1. まず、メッセージプレフィックスを作成します。プレフィックスには、ドメイン区切り文字、コンテキストの長さ、およびコンテキストが含まれます。ドメイン区切り文字とコンテキストの長さのデフォルトは 0 です。
2. メッセージプレフィックスをメッセージに追加します。
3. SHAKE256 を使用してパブリックキーをハッシュし、ステップ 2 の結果に付加します。
4. 最後に、ステップ 3 の結果をハッシュして 64 バイトの μ を生成しますEXTERNAL_MU。

次の例では、OpenSSL 3.5 を使用して μ を構築しますEXTERNAL_MU。

```
{
  openssl asn1parse -inform DER -in public_key.der -strparse 17 -noout -out - 2>/dev/
null |
  openssl dgst -provider default -shake256 -xoflen 64 -binary;
  printf '\x00\x00';
  echo -n "your message"
} | openssl dgst -provider default -shake256 -xoflen 64 -binary > mu.bin
```

mu.bin ファイルを作成したら、次のコマンドを使用して AWS KMS API を呼び出し、メッセージに署名します。

```
aws kms sign \  
  --key-id _<1234abcd-12ab-34cd-56ef-1234567890ab>_ \  
  --message fileb://mu.bin \  
  --message-type EXTERNAL_MU \  
  --signing-algorithm ML_DSA_SHAKE_256 \  
  --output text \  
  --query Signature | base64 --decode > ExampleSignature.bin
```

結果として得られる署名は、元のメッセージRAWの署名と同じです。同じ OpenSSL 3.5 コマンドを使用してメッセージを確認できます。

```
echo -n 'your message' | ./openssl dgst -verify public_key.der -signature  
ExampleSignature.bin
```

SM2 キーペアによるオフライン検証 (中国リージョンのみ)

SM2 パブリックキー AWS KMS を使用して の外部の署名を検証するには、識別 ID を指定する必要があります。未加工のメッセージ を [Sign](#) API [MessageType:RAW](#) に渡すと、AWS KMS は 1234567812345678GM/T 0009-2012 の OSCCA で定義されたデフォルトの識別 ID を使用します。独自の識別 ID を AWS KMS で指定することはできません。

ただし、 の外部でメッセージダイジェストを生成する場合は AWS、独自の識別 ID を指定し、メッセージダイジェスト を [MessageType:DIGEST](#) に渡 AWS KMS して署名できます。これを行うには、SM2OfflineOperationHelper クラスの DEFAULT_DISTINGUISHING_ID 値を変更します。指定する識別 ID は、最大 8,192 文字の任意の文字列です。がメッセージダイジェスト AWS KMS に署名したら、メッセージダイジェストまたはメッセージ、およびダイジェストを計算してオフラインで検証するために使用される識別 ID が必要です。

Important

SM2OfflineOperationHelper リファレンスコードは [Bouncy Castle](#) バージョン 1.68 と互換性があるように設計されています。他のバージョンに関するヘルプについては、[bouncycastle.org](https://www.bouncycastle.org) にアクセスしてください。

SM2OfflineOperationHelper クラス

SM2 キーのオフライン操作の支援向けに、Java の SM2OfflineOperationHelper クラスにはいくつかのタスク実行メソッドが用意されています。このヘルパークラスは、他の暗号化プロバイダのモデルとして使用できます。

内では AWS KMS、raw 暗号文変換と SM2DSA メッセージダイジェスト計算が自動的に行われます。どの暗号化プロバイダーも同じ方法で SM2 を実装しているとは限りません。[OpenSSL](#) バージョン 1.1.1 以降などの一部のライブラリは、これらのアクションを自動的に実行します。は、OpenSSL バージョン 3.0 でのテストでこの動作 AWS KMS を確認しました。変換と計算を手動で実行するには、[Bouncy Castle](#) などのライブラリを持つ以下の SM2OfflineOperationHelper クラスを使用します。

SM2OfflineOperationHelper クラスは、次のオフラインオペレーションのためのメソッドを提供します。

- メッセージダイジェストの計算

オフライン検証に使用できるメッセージダイジェストをオフラインで生成したり、に渡 AWS KMS して署名したりするには、calculateSM2Digest メソッドを使用します。calculateSM2Digest メソッドは SM3 ハッシュアルゴリズムでメッセージダイジェストを生成します。[GetPublicKey](#) API は、パブリックキーをバイナリ形式で返します。バイナリキーを解析して Java PublicKey にする必要があります。解析されたパブリックキーをメッセージとともに提供します。このメソッドは、メッセージをデフォルトの識別 ID、1234567812345678 と自動的に組み合わせますが、DEFAULT_DISTINGUISHING_ID 値を変更して、独自の識別 ID を設定することもできます。

- 検証

署名をオフラインで検証するには、offlineSM2DSAVerify メソッドを使用します。offlineSM2DSAVerify メソッドは、指定された識別 ID から計算されたメッセージダイジェストと、指定された元のメッセージを使用してデジタル署名を検証します。[GetPublicKey](#) API は、パブリックキーをバイナリ形式で返します。バイナリキーを解析して Java PublicKey にする必要があります。解析されたパブリックキーに、元のメッセージと検証する署名を指定します。詳細については、「[SM2 キーペアによるオフライン検証](#)」を参照してください。

- 暗号化

プレーンテキストをオフラインで暗号化するには、offlineSM2PKEEncrypt メソッドを使用します。この方法では、暗号文が復号 AWS KMS できる形式になります。offlineSM2PKEEncrypt メソッドは、プレーンテキストを暗号化し、生成された生の暗

号文を SM2PKE によって ASN.1 形式に変換します。[GetPublicKey](#) API は、パブリックキーをバイナリ形式で返します。バイナリキーを解析して Java PublicKey にする必要があります。解析したパブリックキーに、暗号化するプレーンテキストを指定します。

変換を実行する必要があるかどうか分からない場合は、次の OpenSSL オペレーションを使用して暗号文の形式をテストします。オペレーションが失敗した場合は、暗号文を ASN.1 形式に変換する必要があります。

```
openssl asn1parse -inform DER -in ciphertext.der
```

デフォルトでは、SM2DSA オペレーションのメッセージダイジェストを生成するとき、SM2OfflineOperationHelper クラスはデフォルトの識別 ID、1234567812345678 を使用します。

```
package com.amazon.kms.utils;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import java.io.IOException;
import java.math.BigInteger;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;

import org.bouncycastle.crypto.CryptoException;
import org.bouncycastle.jce.interfaces.ECPublicKey;

import java.util.Arrays;

import org.bouncycastle.asn1.ASN1EncodableVector;
import org.bouncycastle.asn1.ASN1Integer;
import org.bouncycastle.asn1.DEROctetString;
import org.bouncycastle.asn1.DERSequence;
import org.bouncycastle.asn1.gm.GMNamedCurves;
```

```

import org.bouncycastle.asn1.x9.X9ECParameters;
import org.bouncycastle.crypto.CipherParameters;
import org.bouncycastle.crypto.params.ParametersWithID;
import org.bouncycastle.crypto.params.ParametersWithRandom;
import org.bouncycastle.crypto.signers.SM2Signer;
import org.bouncycastle.jcajce.provider.asymmetric.util.ECUtil;

public class SM2OfflineOperationHelper {
    // You can change the DEFAULT_DISTINGUISHING_ID value to set your own
    // distinguishing ID,
    // the DEFAULT_DISTINGUISHING_ID can be any string up to 8,192 characters long.
    private static final byte[] DEFAULT_DISTINGUISHING_ID =
"1234567812345678".getBytes(StandardCharsets.UTF_8);
    private static final X9ECParameters SM2_X9EC_PARAMETERS =
GMNamedCurves.getByName("sm2p256v1");

    // ***calculateSM2Digest***
    // Calculate message digest
    public static byte[] calculateSM2Digest(final PublicKey publicKey, final byte[]
message) throws
        NoSuchProviderException, NoSuchAlgorithmException {
        final ECPublicKey ecPublicKey = (ECPublicKey) publicKey;

        // Generate SM3 hash of default distinguishing ID, 1234567812345678
        final int entlenA = DEFAULT_DISTINGUISHING_ID.length * 8;
        final byte [] entla = new byte[] { (byte) (entlenA & 0xFF00), (byte) (entlenA &
0x00FF) };
        final byte [] a = SM2_X9EC_PARAMETERS.getCurve().getA().getEncoded();
        final byte [] b = SM2_X9EC_PARAMETERS.getCurve().getB().getEncoded();
        final byte [] xg = SM2_X9EC_PARAMETERS.getG().getXCoord().getEncoded();
        final byte [] yg = SM2_X9EC_PARAMETERS.getG().getYCoord().getEncoded();
        final byte[] xa = ecPublicKey.getQ().getXCoord().getEncoded();
        final byte[] ya = ecPublicKey.getQ().getYCoord().getEncoded();
        final byte[] za = MessageDigest.getInstance("SM3", "BC")
            .digest(ByteBuffer.allocate(entla.length +
DEFAULT_DISTINGUISHING_ID.length + a.length + b.length + xg.length + yg.length +
            xa.length +
ya.length).put(entla).put(DEFAULT_DISTINGUISHING_ID).put(a).put(b).put(xg).put(yg).put(xa).put
            .array());

        // Combine hashed distinguishing ID with original message to generate final
        // digest
        return MessageDigest.getInstance("SM3", "BC")

```

```

        .digest(ByteBuffer.allocate(za.length +
message.length).put(za).put(message)
                .array());
    }

    // ***offlineSM2DSAVerify***
    // Verify digital signature with SM2 public key
    public static boolean offlineSM2DSAVerify(final PublicKey publicKey, final byte []
message,
        final byte [] signature) throws InvalidKeyException {
        final SM2Signer signer = new SM2Signer();
        CipherParameters cipherParameters =
ECUtil.generatePublicKeyParameter(publicKey);
        cipherParameters = new ParametersWithID(cipherParameters,
DEFAULT_DISTINGUISHING_ID);
        signer.init(false, cipherParameters);
        signer.update(message, 0, message.length);
        return signer.verifySignature(signature);
    }

    // ***offlineSM2PKEEncrypt***
    // Encrypt data with SM2 public key
    public static byte[] offlineSM2PKEEncrypt(final PublicKey publicKey, final byte []
plaintext) throws
        NoSuchPaddingException, NoSuchAlgorithmException, NoSuchProviderException,
InvalidKeyException,
        BadPaddingException, IllegalBlockSizeException, IOException {
        final Cipher sm2Cipher = Cipher.getInstance("SM2", "BC");
        sm2Cipher.init(Cipher.ENCRYPT_MODE, publicKey);

        // By default, Bouncy Castle returns raw ciphertext in the c1c2c3 format
        final byte [] cipherText = sm2Cipher.doFinal(plaintext);

        // Convert the raw ciphertext to the ASN.1 format before passing it to AWS KMS
        final ASN1EncodableVector asn1EncodableVector = new ASN1EncodableVector();
        final int coordinateLength = (SM2_X9EC_PARAMETERS.getCurve().getFieldSize() +
7) / 8 * 2 + 1;
        final int sm3HashLength = 32;
        final int xCoordinateInCipherText = 33;
        final int yCoordinateInCipherText = 65;
        byte[] coords = new byte[coordinateLength];
        byte[] sm3Hash = new byte[sm3HashLength];
        byte[] remainingCipherText = new byte[cipherText.length - coordinateLength -
sm3HashLength];

```

```
// Split components out of the ciphertext
System.arraycopy(cipherText, 0, coords, 0, coordinateLength);
System.arraycopy(cipherText, cipherText.length - sm3HashLength, sm3Hash, 0,
sm3HashLength);
System.arraycopy(cipherText, coordinateLength, remainingCipherText,
0, cipherText.length - coordinateLength - sm3HashLength);

// Build standard SM2PKE ASN.1 ciphertext vector
asn1EncodableVector.add(new ASN1Integer(new BigInteger(1,
Arrays.copyOfRange(coords, 1, xCoordinateInCipherText))));
asn1EncodableVector.add(new ASN1Integer(new BigInteger(1,
Arrays.copyOfRange(coords, xCoordinateInCipherText, yCoordinateInCipherText))));
asn1EncodableVector.add(new DEROctetString(sm3Hash));
asn1EncodableVector.add(new DEROctetString(remainingCipherText));

return new DERSequence(asn1EncodableVector).getEncoded("DER");
}
}
```

モニタリング AWS KMS keys

モニタリングは、での 可用性、状態、使用状況を理解し、AWS ソリューションの信頼性、可用性、パフォーマンス AWS KMS keys AWS KMS を維持する上で重要な部分です。AWS ソリューションのすべてのパートからモニタリングデータを収集すると、マルチポイント障害が発生した場合にそれをデバッグするのに役立ちます。ただし、KMS キーのモニタリングをスタートする前に、以下の質問に対する回答を反映したモニタリング計画を作成する必要があります。

- モニタリングの目的は何ですか？
- どのリソースをモニタリングしますか？
- どのくらいの頻度でこれらのリソースをモニタリングしますか？
- 使用する [モニタリングツール](#) は？
- 誰がモニタリングタスクを実行しますか？
- 問題が発生したときに誰が通知を受け取りますか？

次のステップでは、KMS キーを経時的にモニタリングして、環境内での通常の AWS KMS の使用および想定ベースラインを確立します。KMS キーをモニタリングする際に、過去のモニタリングデータを保存し、現在のデータと比較することで、通常パターンと異常パターンを識別して、問題に対処する方法を考案できるようにします。

例えば、KMS キーに影響する AWS KMS API アクティビティとイベントをモニタリングできます。データが、確立基準を上回ったり、下回ったりする場合、修正作業を調査、または実行する必要がある場合があります。

通常のパターンのベースラインを確立するには、次の項目をモニタリングします。

- AWS KMS データプレーンオペレーションの API アクティビティ。これは、KMS キーを使用する [暗号化オペレーション](#) ([Decrypt](#)、[Encrypt](#)、[ReEncrypt](#)、[GenerateDataKey](#) など) です。
- AWS KMS ユーザーにとって重要なコントロールプレーンオペレーションの API アクティビティ。これらのオペレーションは KMS キーを管理し、KMS キーの可用性を変更するオペレーション ([ScheduleKeyDeletion](#)、[CancelKeyDeletion](#)、[DisableKey](#)、[EnableKey](#)、[ImportKeyMaterial](#)、[DeleteImport](#) など) をモニタリングします。または、KMS キーのアクセスコントロール ([PutKeyPolicy](#) および [RevokeGrant](#) など) を変更します。

- その他の AWS KMS メトリクス ([インポートされたキーマテリアル](#)の有効期限が切れるまでの残り時間など) とイベント (インポートされたキーマテリアルの有効期限、KMS キーの削除またはキーローテーションなど)。

モニタリングツール

AWS には、KMS キーのモニタリングに使用できるさまざまなツールが用意されています。これらのツールの一部はモニタリングを行うように設定できますが、一部のツールは手動による介入が必要です。モニタリングタスクをできるだけ自動化することをお勧めします。

自動モニタリングツール

次の自動化されたモニタリングツールを使用して KMS キーを監視し、変更が生じたときに報告させることができます。

- AWS CloudTrail ログモニタリング – アカウント間でログファイルを共有し、CloudWatch Logs に送信CloudWatch CloudTrail ログファイルをリアルタイムでモニタリングし、[CloudTrail Processing Library](#) を使用してログ処理アプリケーションを書き込み、CloudTrail による配信後にログファイルが変更されていないことを確認します。詳細については、AWS CloudTrail ユーザーガイドの「[CloudTrail ログファイルのオペレーション](#)」を参照してください。
- Amazon CloudWatch アラーム - 指定した期間にわたって単一のメトリクスをモニタリングし、複数の期間にわたる特定のしきい値に対するメトリクスの値に基づいて 1 つ以上のアクションを実行します。アクションは、Amazon Simple Notification Service (Amazon SNS) のトピックまたは Amazon EC2 Auto Scaling のポリシーに送信される通知です。CloudWatch アラームは、特定の状態にあるという理由だけでアクションを呼び出すことはありません。状態が変更され、指定された期間維持されている必要があります。詳細については、「[Amazon CloudWatch で KMS キーをモニタリングする](#)」を参照してください。
- Amazon EventBridge — イベントを照合し、1 つ以上のターゲット関数またはストリームにルーティングして、状態情報をキャプチャし、必要に応じて変更または修正措置を講じます。詳細については、[Amazon EventBridge を使用した KMS キーのモニタリング](#) および「[Amazon EventBridge ユーザーガイド](#)」を参照してください。
- Amazon CloudWatch Logs – AWS CloudTrail またはその他のソースからのログファイルをモニタリング、保存、およびアクセスします。詳細については、「[Amazon CloudWatch Logs ユーザーガイド](#)」を参照してください。

手動モニタリングツール

KMS キーのモニタリングにおいてもう 1 つの重要な部分は、CloudWatch アラームおよびイベントでカバーされない項目を手動でモニタリングすることです。AWS KMS、CloudWatch AWS Trusted Advisor、およびその他の AWS ダッシュボードには、環境の状態が at-a-glance ビューが表示されます AWS。

[AWS マネージドキー] および [AWS KMS コンソール](#) の [Customer managed keys] (カスタマー マネージドキー) ページを [カスタマイズ](#) して、各 KMS キーに関する次の情報を表示できます。

- キー ID
- ステータス
- 作成日
- 有効期限 ([インポートされたキーマテリアル](#) を持つ KMS キーの場合)
- オリジン
- カスタムキーストア ID ([カスタムキーストア](#) の KMS キーの場合)

[CloudWatch コンソールのダッシュボード](#) には、以下が表示されます。

- 現在のアラームとステータス
- アラームとリソースのグラフ
- サービスのヘルスステータス

また、CloudWatch を使用して以下のことを行えます。

- 重視するサービスをモニタリングするための [カスタマイズしたダッシュボード](#) を作成します
- メトリクスデータをグラフ化して、問題のトラブルシューティングを行い、傾向を確認する
- すべての AWS リソースメトリクスを検索して参照する
- 問題があることを通知するアラームを作成/編集する

AWS Trusted Advisor は、AWS リソースをモニタリングして、パフォーマンス、信頼性、セキュリティ、コスト効率を向上させるのに役立ちます。すべてのユーザーが 4 つの Trusted Advisor チェックを利用できます。ビジネスまたはエンタープライズサポートプランのユーザーは 50 を超えるチェックを利用できます。詳細については、「[AWS Trusted Advisor](#)」を参照してください。

を使用した AWS KMS API コールのログ記録 AWS CloudTrail

AWS KMS は、ユーザー [AWS CloudTrail](#)、ロール、およびその他の サービス AWS KMS による へのすべての呼び出しを記録する AWS サービスであると統合されています。CloudTrail は、AWS KMS コンソール、API、AWS CloudFormation テンプレート、AWS Command Line Interface (AWS CLI)、からの呼び出しなど、へのすべての AWS KMS APIs 呼び出しをイベント AWS KMS としてキャプチャします AWS Tools for PowerShell。

CloudTrail は AWS KMS、[ListAliases](#) や [GetKeyRotationStatus](#) などの読み取り専用オペレーション、[CreateKey](#) や [PutKeyPolicy](#) などの KMS キーを管理するオペレーション、[GenerateDataKey](#) や [Decrypt](#) などの [暗号化オペレーションを含むすべてのオペレーション](#) をログに記録します。また、[DeleteExpiredKeyMaterial](#)、[DeleteKey](#)、[SynchronizeMultiRegionKey](#)、[RotateKey](#) など、が AWS KMS 呼び出す内部オペレーションもログに記録されます。

CloudTrail は、すべての成功したオペレーションと、呼び出し元がリソースへのアクセスを拒否されたときなど、失敗した試行された呼び出しをログに記録します。[KMS キーに対するアカウントを横断したオペレーション](#) は、発信者のアカウントと KMS キー所有者のアカウントの両方に記録されます。ただし、アクセスが拒否されたために拒否されたクロスアカウント AWS KMS リクエストは、発信者のアカウントにのみ記録されます。

セキュリティ上の理由から、[Encrypt](#) リクエストの Plaintext パラメータ、[GetKeyPolicy](#) または暗号化オペレーションへのレスポンスなど、一部のフィールドは AWS KMS ログエントリから省略されます。特定の KMS キーの CloudTrail ログエントリを検索しやすくするために、API オペレーションが [キー ARN](#) を返さない場合でも、は影響を受ける KMS キーのキー ARN を一部の AWS KMS キー管理オペレーションのログエントリの responseElements フィールド AWS KMS に追加します。

デフォルトでは、すべての AWS KMS アクションは CloudTrail イベントとして記録されますが、CloudTrail 証跡からアクションを除外 AWS KMS できます。詳細については、「[証跡からの AWS KMS イベントの除外](#)」を参照してください。

詳細はこちら:

- AWS Nitro Enclave の AWS KMS オペレーションの CloudTrail ログの例については、「」を参照してください [Nitro Enclaves に対するリクエストの監視](#)。

トピック

- [CloudTrail での AWS KMS ログエントリの検索](#)

- [証跡からの AWS KMS イベントの除外](#)
- [AWS KMS ログエントリの例](#)

CloudTrail での AWS KMS ログエントリの検索

CloudTrail ログエントリを検索するときは、[CloudTrail コンソール](#)か [CloudTrail LookupEvents](#) オペレーションを使用します。CloudTrail は、イベント名、ユーザー名、イベントソースなど、検索をフィルタリングするための多数の[属性値](#)をサポートしています。

CloudTrail で AWS KMS ログエントリを検索しやすくするために、AWS KMS は次の CloudTrail ログエントリフィールドに入力します。

Note

2022 年 12 月以降、は特定の KMS キーを変更するすべての管理オペレーションでリソースタイプとリソース名属性 AWS KMS を入力します。これらの属性値は、次のオペレーションでは、NULL が、CloudTrail の古いエントリになる場合があります：

[CreateAlias](#)、[CreateGrant](#)、[DeleteAlias](#)、[DeleteImportedKeyMaterial](#)、[ImportKeyMaterial](#)、[ReplicateKey](#)

属性	値	ログエントリ
イベントソース (EventSource)	kms.amazonaws.com	すべてのオペレーション
リソースタイプ (ResourceType)	AWS::KMS::Key	特定の KMS キーを変更する管理オペレーション (CreateKey や EnableKey など。ListKeys は除く)。
リソース名 (ResourceName)	キー ARN (またはキー ID およびキー ARN)	特定の KMS キーを変更する管理オペレーション (CreateKey や EnableKey など。ListKeys は除く)。

特定の KMS キーに対する管理オペレーションのログエントリを検索しやすくするために、AWS KMS は、AWS KMS API オペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN をログエントリの `responseElements.keyId` 要素に記録します。

例えば、[DisableKey](#) オペレーションの呼び出しが成功しても応答には値が返されませんが、NULL 値の代わりに、[DisableKey ログエントリ](#) の `responseElements.keyId` 値には、無効になっている KMS キーのキー ARN が追加されます。

この機能は 2022 年 12 月に追加されました。その影響を受ける CloudTrail ログエントリは以下の通りです:

[CreateAlias](#)、[CreateGrant](#)、[DeleteAlias](#)、[DeleteKey](#)、[DisableKey](#)、[EnableKey](#)、[EnableKeyRotation](#)、[ImportKeyMaterial](#)

証跡からの AWS KMS イベントの除外

AWS KMS リソースの使用と管理の記録を提供するために、ほとんどの AWS KMS ユーザーは CloudTrail 証跡のイベントに依存しています。証跡は、キーポリシーの作成、無効化、削除 AWS KMS keys、変更、AWS サービスによる KMS キーの使用など、重要なイベントを監査するための貴重なデータソースになります。場合によっては、CloudTrail ログエントリのメタデータ (暗号化オペレーションの[暗号化コンテキスト](#)など) が、エラーを回避または解決するために役立ちます。

ただし、AWS KMS は多数のイベントを生成できるため、証跡から AWS KMS イベントを除外 AWS CloudTrail します。この証跡ごとの設定では、すべての AWS KMS イベントを除外します。特定の AWS KMS イベントを除外することはできません。

Warning

CloudTrail ログから AWS KMS イベントを除外すると、KMS キーを使用するアクションが隠される可能性があります。このオペレーションを実行するために必要な `cloudtrail:PutEventSelectors` アクセス許可をプリンシパルに与えるときは注意してください。

証跡から AWS KMS イベントを除外するには：

- CloudTrail コンソールでは、[追跡の作成時](#)または[追跡の更新時](#)に、[Log Key Management Service events] (Key Management Service のイベントを記録) 設定を使用します。手順については、「[AWS CloudTrail ユーザーガイド](#)」の「[を使用した管理イベントのログ記録 AWS Management Console](#)」を参照してください。

- CloudTrail API では、[PutEventSelectors](#) オペレーションを実行します。ExcludeManagementEventSources 属性を kms.amazonaws.com の値でイベントセレクトタに追加します。例については、「AWS CloudTrail ユーザーガイド」の「[例: AWS Key Management Service イベントをログに記録しない証跡](#)」を参照してください。

この除外は、コンソール設定または証跡のイベントセレクトタを変更することでいつでも無効にできます。その後、証跡は AWS KMS イベントの記録を開始します。ただし、除外が有効である間に発生した AWS KMS イベントを回復することはできません。

コンソールまたは API を使用して AWS KMS イベントを除外すると、結果の CloudTrail PutEventSelectors API オペレーションも CloudTrail Logs に記録されます。AWS KMS イベントが CloudTrail ログに表示されない場合は、ExcludeManagementEventSources 属性が kms.amazonaws.com に設定されている PutEventSelectors イベントを探します。

AWS KMS ログエントリの例

AWS KMS は、AWS KMS オペレーションを呼び出すとき、および AWS サービスがユーザーに代わって オペレーションを呼び出すときに CloudTrail ログにエントリを書き込みます。AWS KMS また、は、 オペレーションを呼び出すときにもエントリを書き込みます。例えば、削除をスケジュールした [KMS キーを削除する](#) と、エントリが書き込まれます。

以下のトピックでは、AWS KMS オペレーションの CloudTrail ログエントリの例を示します。

AWS Nitro Enclaves AWS KMS から へのリクエストの CloudTrail ログエントリの例については、「」を参照してください[Nitro Enclaves に対するリクエストの監視](#)。

トピック

- [CancelKeyDeletion](#)
- [ConnectCustomKeyStore](#)
- [CreateAlias](#)
- [CreateCustomKeyStore](#)
- [CreateGrant](#)
- [CreateKey](#)
- [Decrypt](#)
- [DeleteAlias](#)
- [DeleteCustomKeyStore](#)

- [期限切れのキーのマテリアルを削除](#)
- [DeleteImportedKeyMaterial](#)
- [DeleteKey](#)
- [DescribeCustomKeyStores](#)
- [DescribeKey](#)
- [DisableKey](#)
- [DisableKeyRotation](#)
- [DisconnectCustomKeyStore](#)
- [EnableKey](#)
- [EnableKeyRotation](#)
- [暗号化](#)
- [GenerateDataKey](#)
- [GenerateDataKeyPair](#)
- [GenerateDataKeyPairWithoutPlaintext](#)
- [GenerateDataKeyWithoutPlaintext](#)
- [GenerateMac](#)
- [GenerateRandom](#)
- [GetKeyPolicy](#)
- [GetKeyRotationStatus](#)
- [GetParametersForImport](#)
- [ImportKeyMaterial](#)
- [ListAliases](#)
- [ListGrants](#)
- [ListKeyRotations](#)
- [PutKeyPolicy](#)
- [ReEncrypt](#)
- [ReplicateKey](#)
- [RetireGrant](#)
- [RevokeGrant](#)

- [RotateKey](#)
- [RotateKeyOnDemand](#)
- [ScheduleKeyDeletion](#)
- [Sign](#)
- [SynchronizeMultiRegionKey](#)
- [TagResource](#)
- [UntagResource](#)
- [UpdateAlias](#)
- [UpdateCustomKeyStore](#)
- [UpdateKeyDescription](#)
- [UpdatePrimaryRegion](#)
- [VerifyMac](#)
- [Verify](#)
- [Amazon EC2 の例1](#)
- [Amazon EC2 の例 2](#)

CancelKeyDeletion

次の例は、[CancelKeyDeletion](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。削除の詳細については AWS KMS keys、「」を参照してください [を削除する AWS KMS key](#)。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2020-07-27T21:53:17Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CancelKeyDeletion",
  "awsRegion": "us-west-2",
```



```

    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "e3452e68-d4b0-4ec7-a768-7ae96c23764f",
    "eventID": "d818bf03-6655-48e9-8b26-f279a07075fd",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}

```

ConnectCustomKeyStore

次の例は、[ConnectCustomKeyStore](#)オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。カスタムキーストアの接続に関する詳細については、「[AWS CloudHSM キーストアを切断する](#)」を参照してください。

```

{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-21T20:17:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ConnectCustomKeyStore",

```

```
{
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "customKeyStoreId": "cks-1234567890abcdef0"
  },
  "responseElements": null,
  "additionalEventData": {
    "customKeyStoreName": "ExampleKeyStore",
    "clusterId": "cluster-1a23b4cdefg"
  },
  "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
  "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
}
```

CreateAlias

次の例は、[CreateAlias](#) オペレーションの AWS CloudTrail ログエントリを示しています。resources 要素には、エイリアスと KMS キーリソースのフィールドが含まれます。でのエイリアスの作成については AWS KMS、「」を参照してください[エイリアスを作成する](#)。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が responseElements.keyId 値に含まれます。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-08-14T23:08:31Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateAlias",
  "awsRegion": "us-west-2",
```

```

    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "aliasName": "alias/ExampleAlias",
        "targetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "caec1e0c-ce03-419e-bdab-6ab1f7c57c01",
    "eventID": "2dd6e784-8286-46a6-befd-d64e5a02fb28",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}

```

CreateCustomKeyStore

次の例は、AWS CloudHSM キーストアで [CreateCustomKeyStore](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。カスタムキーストアの作成に関する詳細については、「[AWS CloudHSM キーストアを作成する](#)」を参照してください。

```

{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",

```

```

    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-10-21T20:17:32Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateCustomKeyStore",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "customKeyStoreName": "ExampleKeyStore",
    "clusterId": "cluster-1a23b4cdefg"
  },
  "responseElements": {
    "customKeyStoreId": "cks-1234567890abcdef0"
  },
  "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
  "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
}

```

CreateGrant

次の例は、[CreateGrant](#) オペレーションの AWS CloudTrail ログエントリを示しています。で権限を作成する方法については AWS KMS、「」を参照してくださいの[許可 AWS KMS](#)。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

```

{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",

```

```

    "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:53:12Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateGrant",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "constraints": {
      "encryptionContextSubset": {
        "ContextKey1": "Value1"
      }
    },
    "operations": ["Encrypt",
"RetireGrant"],
    "granteePrincipal": "EX_PRINCIPAL_ID"
  },
  "responseElements": {
    "grantId": "f020fe75197b93991dc8491d6f19dd3cebb24ee62277a05914386724f3d48758",
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "f3c08808-63bc-11e4-bc2b-4198b6150d5c",
  "eventID": "5d529779-2d27-42b5-92da-91aaea1fc4b5",
  "readOnly": false,
  "resources": [{
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}

```

CreateKey

これらの例は、[CreateKey](#) オペレーションの AWS CloudTrail ログエントリを示しています。

CreateKey ログエントリは、[ReplicateKey](#) リクエストの CreateKey リクエストまたは CreateKey オペレーションによって生成されます。

以下は、[対称暗号化 KMS キー](#)を作成する [CreateKey](#) オペレーションの CloudTrail ログエントリを示す例です。KMS キー作成の詳細については、[KMS キーを作成する](#) を参照してください。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-08-10T22:38:27Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "description": "",
    "origin": "EXTERNAL",
    "bypassPolicyLockoutSafetyCheck": false,
    "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "keySpec": "SYMMETRIC_DEFAULT",
    "keyUsage": "ENCRYPT_DECRYPT"
  },
  "responseElements": {
    "keyMetadata": {
      "AWSAccountId": "111122223333",
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "creationDate": "Aug 10, 2022, 10:38:27 PM",
      "enabled": false,
      "description": "",
      "keyUsage": "ENCRYPT_DECRYPT",
      "keyState": "PendingImport",
      "origin": "EXTERNAL",
      "keyManager": "CUSTOMER",
      "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
      "keySpec": "SYMMETRIC_DEFAULT",
      "encryptionAlgorithms": [
```

```

        "SYMMETRIC_DEFAULT"
      ],
      "multiRegion": false
    }
  ],
  "requestID": "1aef6713-0223-4ff7-9a6d-781360521930",
  "eventID": "36327b37-f4f6-40a9-92ab-48064ec905a2",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}

```

次の例は、[AWS CloudHSM キーストア](#)に対称暗号化 KMS キーを作成する CreateKey オペレーションの CloudTrail ログを示しています。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-10-14T17:39:50Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyUsage": "ENCRYPT_DECRYPT",

```

```

    "bypassPolicyLockoutSafetyCheck": false,
    "origin": "AWS_CLOUDHSM",
    "keySpec": "SYMMETRIC_DEFAULT",
    "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "customKeyStoreId": "cks-1234567890abcdef0",
    "description": ""
  },
  "responseElements": {
    "keyMetadata": {
      "awsAccountId": "111122223333",
      "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
      "arn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
      "creationDate": "Oct 14, 2021, 5:39:50 PM",
      "enabled": true,
      "description": "",
      "keyUsage": "ENCRYPT_DECRYPT",
      "keyState": "Enabled",
      "origin": "AWS_CLOUDHSM",
      "customKeyStoreId": "cks-1234567890abcdef0",
      "cloudHsmClusterId": "cluster-1a23b4cdefg",
      "keyManager": "CUSTOMER",
      "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
      "keySpec": "SYMMETRIC_DEFAULT",
      "encryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
      ],
      "multiRegion": false
    }
  },
  "additionalEventData": {
    "backingKey": "{\"backingKeyId\": \"backing-key-id\"}"
  },
  "requestID": "4f0b185c-588c-4767-9e90-c618f7e13cad",
  "eventID": "c73964b8-703d-49e4-bd9e-f773d0ee1e65",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
    }
  ],

```



```
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}
```

次の例は、[外部キーストア](#)に对称暗号化 KMS キーを作成する CreateKey オペレーションの CloudTrail ログを示しています。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-09-07T22:37:45Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "CreateKey",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "tags": [],
    "keyUsage": "ENCRYPT_DECRYPT",
    "description": "",
    "origin": "EXTERNAL_KEY_STORE",
    "multiRegion": false,
    "keySpec": "SYMMETRIC_DEFAULT",
    "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "bypassPolicyLockoutSafetyCheck": false,
    "customKeyStoreId": "cks-1234567890abcdef0",
    "xksKeyId": "bb8562717f809024"
  },
  "responseElements": {
    "keyMetadata": {
      "awsAccountId": "111122223333",
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",

```

```

        "creationDate": "Dec 7, 2022, 10:37:45 PM",
        "enabled": true,
        "description": "",
        "keyUsage": "ENCRYPT_DECRYPT",
        "keyState": "Enabled",
        "origin": "EXTERNAL_KEY_STORE",
        "customKeyStoreId": "cks-1234567890abcdef0",
        "keyManager": "CUSTOMER",
        "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "keySpec": "SYMMETRIC_DEFAULT",
        "encryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "multiRegion": false,
        "xksKeyConfiguration": {
            "id": "bb8562717f809024"
        }
    },
    "requestID": "ba197c82-3ac7-487a-8ff4-7736bbbeb1316",
    "eventID": "838ad5f4-5fdd-4044-afd7-4dbd88c6af56",
    "readOnly": false,
    "resources": [
        {
            "accountId": "227179770375",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-east-1:227179770375:key/39c5eb22-f37c-4956-92ca-89e8f8b57ab2"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}

```

Decrypt

これらの例は、[Decrypt](#) オペレーションの AWS CloudTrail ログエントリを示しています。

Decrypt オペレーションの CloudTrail ログエントリには、リクエストで暗号化アルゴリズムが指定されていなくても、常に `requestParameters` に `encryptionAlgorithm` が含まれます。リクエスト内の暗号化テキストとレスポンス内のプレーンテキストは省略されます。

トピック

- [標準の対称暗号化キーを使用した復号](#)
- [標準の対称暗号化キーを使用した復号の失敗](#)
- [キーストアで KMS AWS CloudHSM キーを使用して復号する](#)
- [外部キーストアの KMS キーを使用した復号](#)
- [外部キーストアの KMS キーを使用した復号の失敗](#)

標準の対称暗号化キーを使用した復号

次の CloudTrail ログエントリの例では、Decrypt オペレーションに標準の対称暗号化キーを使用しています。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2025-05-20T20:45:00Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "encryptionContext": {
      "Department": "Engineering",
      "Project": "Alpha"
    }
  },
  "responseElements": null,
  "additionalEventData": {
    "keyMaterialId":
      "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  }
}
```

```

    },
    "requestID": "12345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.2",
      "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
      "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
  }
}

```

標準の対称暗号化キーを使用した復号の失敗

次の CloudTrail ログエントリの例では、標準の対称暗号化 KMS キーを使用した、失敗した Decrypt オペレーションが記録されています。例外 (errorCode) とエラーメッセージ (errorMessage) が記載されており、エラーの解決に役立ちます。

このケースでは、Decrypt リクエストで指定された対称暗号化 KMS キーは、データの暗号化に使用された対称暗号化 KMS キーではありませんでした。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-11-24T18:57:43Z",

```

```

    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "errorCode": "IncorrectKeyException"
    "errorMessage": "The key ID in the request does not identify a CMK that can perform
this operation.",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "encryptionContext": {
            "Department": "Engineering",
            "Project": "Alpha"
        }
    },
    "responseElements": null,
    "requestID": "22345126-30d5-4b28-98b9-9153da559963",
    "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}

```

キーストアで KMS AWS CloudHSM キーを使用して復号する

次の CloudTrail ログエントリの例では、[AWS CloudHSM キーストア](#)の KMS キーを使用した Decrypt オペレーションが記録されています。カスタムキーストアの KMS キーを使用した暗号化オペレーションのすべてのログエントリには、customKeyId と backingKeyId を持つ additionalEventData フィールドが含まれます。backingKeyId フィールドで返される値は、CloudHSM キーの id 属性です。additionalEventData は、このリクエストでは指定されていません。

```
{
```

```
"eventVersion": "1.08",
"userIdentity": {
  "type": "IAMUser",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::111122223333:user/Alice",
  "accountId": "111122223333",
  "accessKeyId": "EXAMPLE_KEY_ID",
  "userName": "Alice"
},
"eventTime": "2021-10-26T23:41:27Z",
"eventSource": "kms.amazonaws.com",
"eventName": "Decrypt",
"awsRegion": "us-west-2",
"sourceIPAddress": "192.0.2.0",
"requestParameters": {
  "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
  "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "encryptionContext": {
    "Department": "Development",
    "Purpose": "Test"
  }
},
"responseElements": null,
"additionalEventData": {
  "customKeyId": "cks-1234567890abcdef0"
},
"requestID": "e1b881f8-2048-41f8-b6cc-382b7857ec61",
"eventID": "a79603d5-4cde-46fc-819c-a7cf547b9df4",
"readOnly": true,
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}
```

外部キーストアの KMS キーを使用した復号

次の CloudTrail ログエントリの例では、[外部キーストア](#)の KMS キーを使用した Decrypt オペレーションが記録されています。additionalEventData フィールドには、customKeyStoreId の他に[外部キー ID](#) (XksKeyId) が含まれます。additionalEventData は、このリクエストでは指定されていません。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-11-24T00:26:58Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "AWS Internal",
  "requestParameters": {
    "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
    "encryptionContext": {
      "Department": "Engineering",
      "Purpose": "Test"
    }
  },
  "responseElements": null,
  "additionalEventData": {
    "customKeyStoreId": "cks-9876543210fedcba9",
    "xksKeyId": "abc01234567890fe"
  },
  "requestID": "f1b881f8-2048-41f8-b6cc-382b7857ec61",
  "eventID": "b79603d5-4cde-46fc-819c-a7cf547b9df4",
  "readOnly": true,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
```

```

        "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }
}

```

外部キーストアの KMS キーを使用した復号の失敗

次の CloudTrail ログエントリの例では、[外部キーストア](#)の KMS キーを使用した Decrypt オペレーションの失敗が記録されています。CloudWatch は、成功したリクエストだけでなく失敗したリクエストも記録します。失敗を記録するとき、CloudTrail のログエントリには、例外 (errorCode) とそれに伴うエラーメッセージ (errorMessage) が追加されます。

こちらの例のように、失敗リクエストが外部キーストアのプロキシに到達した場合、requestId 値を使って失敗したリクエストを、外部キーストアプロキシが記録する、対応するリクエスト (プロキシが提供している場合) に関連付けることができます。

外部キーストアでの Decrypt リクエストに関するヘルプは、「[復号エラー](#)」を参照してください。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-11-24T00:26:58Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "errorCode": "KMSInvalidStateException",
  "errorMessage": "The external key store proxy rejected the request because the specified ciphertext or additional authenticated data is corrupted, missing, or otherwise invalid.",
}

```



```

    "requestParameters": {
      "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
      "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
      "encryptionContext": {
        "Department": "Engineering",
        "Purpose": "Test"
      }
    },
    "responseElements": null,
    "additionalEventData": {
      "customKeyStoreId": "cks-9876543210fedcba9",
      "xksKeyId": "abc01234567890fe"
    },
    "requestID": "f1b881f8-2048-41f8-b6cc-382b7857ec61",
    "eventID": "b79603d5-4cde-46fc-819c-a7cf547b9df4",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }
}

```

DeleteAlias

次の例は、[DeleteAlias](#) オペレーションの AWS CloudTrail ログエントリを示しています。エイリアスの削除については、を参照してください [エイリアスを削除する](#)。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

```

{
  "eventVersion": "1.02",
  "userIdentity": {

```

```

    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2014-11-04T00:52:27Z"
      }
    }
  },
  "eventTime": "2014-11-04T00:52:27Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DeleteAlias",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "aliasName": "alias/my_alias"
  },
  "responseElements": {
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "d9542792-63bc-11e4-bc2b-4198b6150d5c",
  "eventID": "12f48554-bb04-4991-9cfc-e7e85f68eda0",
  "readOnly": false,
  "resources": [{
    "ARN": "arn:aws:kms:us-east-1:111122223333:alias/my_alias",
    "accountId": "111122223333"
  },
  {
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}

```

DeleteCustomKeyStore

次の例は、[DeleteCustomKeyStore](#)オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。カスタムキーストアの作成に関する詳細については、「[AWS CloudHSM キーストアを削除する](#)」を参照してください。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-10-21T20:17:32Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DeleteCustomKeyStore",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "customKeyStoreId": "cks-1234567890abcdef0"
  },
  "responseElements": null,
  "additionalEventData": {
    "customKeyStoreName": "ExampleKeyStore",
    "clusterId": "cluster-1a23b4cdefg"
  },
  "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
  "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
}
```

期限切れのキーのマテリアルを削除

キーマテリアルを AWS KMS key (KMS キー) にインポートするとき、そのキーマテリアルの有効期限日時を設定できます。は、[キーマテリアルをインポート](#)するとき (有効期限設定を使用)、お

よび が期限切れのキーマテリアル AWS KMS を削除するときに、CloudTrail ログにエントリ AWS KMS を記録します。インポートされたキーマテリアルを使用する KMS キーの作成の詳細については、[キーの AWS KMS キーマテリアルのインポート](#) を参照してください。

次の例は、 が期限切れのキーマテリアル AWS KMS を削除したときに生成される AWS CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2025-05-22T19:55:11Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DeleteExpiredKeyMaterial",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "cfa932fd-0d3a-4a76-a8b8-616863a2b547",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsServiceEvent",
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "keyMaterialId": "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  },
  "eventCategory": "Management"
}
```

DeleteImportedKeyMaterial

KMS キーにキーマテリアルをインポートすると、インポートしたキーマテリアルは、[DeleteImportedKeyMaterial](#) オペレーションを使っていつでも削除できます。インポートしたキーマテリアルを KMS キーから削除すると、この KMS キーのキーの状態は PendingImport に変わり、暗号化オペレーションに使用できなくなります。詳細については、「[インポートされたキーマテリアルを削除する](#)」を参照してください。

次の例は、DeleteImportedKeyMaterial オペレーション用に生成された AWS CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2025-05-20T20:45:08Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DeleteImportedKeyMaterial",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "keyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  },
  "responseElements": {
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "keyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  },
  "requestID": "dcf0e82f-dad0-4622-a378-a5b964ad42c1",
  "eventID": "2afbb991-c668-4641-8a00-67d62e1fecbd",
  "readOnly": false,
  "resources": [
    {
```

```
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
```

DeleteKey

これらの例は、KMS キーが削除されたときに生成される AWS CloudTrail ログエントリを示しています。KMS キーを削除するには、[ScheduleKeyDeletion](#) オペレーションを使用します。指定された待機期間が経過すると、AWS KMS は KMS キーを削除し、次のようなエントリを CloudTrail ログに記録してそのイベントを記録します。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

ScheduleKeyDeletion オペレーションの CloudTrail ログエントリの例については、「[ScheduleKeyDeletion](#)」を参照してください。KMS キーの削除の詳細については、[を削除する AWS KMS key](#) を参照してください。

次の例の CloudTrail ログエントリでは、AWS KMS でキーマテリアルを持つ KMS キーの DeleteKey オペレーションが記録されています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "accountId": "111122223333",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2020-07-31T00:07:00Z",
```

```

    "eventSource": "kms.amazonaws.com",
    "eventName": "DeleteKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "b25f9cda-74e1-4458-847b-4972a0bf9668",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management"
  }

```

次の CloudTrail ログエントリは、AWS CloudHSM [カスタムキーストアの KMS キー](#)のDeleteKeyオペレーションを記録します。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2021-10-26T23:41:27Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DeleteKey",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": {
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },

```

```

    "additionalEventData": {
      "customKeyId": "cks-1234567890abcdef0",
      "clusterId": "cluster-1a23b4cdefg",
      "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]",
      "backingKeysDeletionStatus": "[{\"backingKeyId\":\"backing-key-id\",
\"deletionStatus\":\"SUCCESS\"}]\"",
    },
    "eventID": "1234585c-4b0c-4340-ab11-662414b79239",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management"
  }
}

```

DescribeCustomKeyStores

次の例は、[DescribeCustomKeyStores](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。カスタムキーストアの表示に関する詳細については、「[AWS CloudHSM キーストアを表示する](#)」を参照してください。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-10-21T20:17:32Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DescribeCustomKeyStores",
  "awsRegion": "us-east-1",

```



```
{
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "customKeyStoreId": "cks-1234567890abcdef0"
  },
  "responseElements": null,
  "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
  "eventID": "2ea1735f-628d-43e3-b2ee-486d02913a78",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
}
```

DescribeKey

次の例は、[DescribeKey](#) オペレーションの AWS CloudTrail ログエントリを示しています。は、[DescribeKey](#) オペレーションを呼び出すか、AWS KMS コンソールで [KMS キーを表示する](#) ときに、次のようなエントリ AWS KMS を記録します。この呼び出しは、AWS KMS マネジメントコンソールでキーを表示した結果です。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-09-26T18:01:36Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DescribeKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": null,
  "requestID": "12345126-30d5-4b28-98b9-9153da559963",
  "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
}
```

```
{
  "readOnly": true,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

DisableKey

次の例は、[DisableKey](#) オペレーションの AWS CloudTrail ログエントリを示しています。AWS KMS keys での有効化と無効化の詳細については AWS KMS、「」を参照してください [キーの有効化と無効化](#)。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

```
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:52:43Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DisableKey",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": {
```

```

    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "12345126-30d5-4b28-98b9-9153da559963",
  "eventID": "abcde202-ba1a-467c-b4ba-f729d45ae521",
  "readOnly": false,
  "resources": [{
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}

```

DisableKeyRotation

次の例は、[DisableKeyRotation](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。自動キーローテーションについては、「[ローテーション AWS KMS keys](#)」を参照してください。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-09-01T19:31:39Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DisableKeyRotation",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": null,
  "requestID": "d6a9351a-ed6e-4581-88d1-2a9a8a538497",

```

```

    "eventID": "6313164c-83aa-4cc3-9e1a-b7c426f7a5b1",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }
}

```

DisconnectCustomKeyStore

次の例は、[DisconnectCustomKeyStore](#)オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。カスタムキーストアの切断に関する詳細については、「[AWS CloudHSM キーストアを切断する](#)」を参照してください。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-10-21T20:17:32Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DisconnectCustomKeyStore",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "customKeyId": "cks-1234567890abcdef0"
  },
  "responseElements": null,
  "additionalEventData": {

```

```

    "customKeyStoreName": "ExampleKeyStore",
    "clusterId": "cluster-1a23b4cdefg"
  },
  "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
  "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
}

```

EnableKey

次の例は、[EnableKey](#) オペレーションの AWS CloudTrail ログエントリを示しています。AWS KMS keys での有効化と無効化の詳細については AWS KMS、「」を参照してください [キーの有効化と無効化](#)。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

```

{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:52:20Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "EnableKey",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": {
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
}

```

```
{,
  "requestID": "d528a6fb-63bc-11e4-bc2b-4198b6150d5c",
  "eventID": "be393928-3629-4370-9634-567f9274d52e",
  "readOnly": false,
  "resources": [{
    "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

EnableKeyRotation

次の例は、[EnableKeyRotation](#) オペレーションへの呼び出しの AWS CloudTrail ログエントリを示しています。キーがローテーションされたときに書き込まれる CloudTrail ログエントリの例については、「[RotateKey](#)」を参照してください。AWS KMS keysのローテーションの詳細については、[ローテーション AWS KMS keys](#) を参照してください。

Note

[rotation-period](#) はオプションのリクエストパラメータです。自動キーローテーションを有効にするとときにローテーション期間を指定しない場合、デフォルト値は 365 日となります。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2020-07-25T23:41:56Z",
```

```

    "eventSource": "kms.amazonaws.com",
    "eventName": "EnableKeyRotation",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "rotationPeriodInDays": 180
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "81f5b794-452b-4d6a-932b-68c188165273",
    "eventID": "fefc43a7-8e06-419f-bcab-b3bf18d6a401",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}

```

暗号化

次の例は、[暗号化](#)オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",

```

```

    "eventSource": "kms.amazonaws.com",
    "eventName": "Encrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "encryptionContext": {
        "Department": "Engineering"
      },
      "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    },
    "responseElements": null,
    "additionalEventData": {
      "keyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "f3423043-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "91235988-eb87-476a-ac2c-0cdc244e6dca",
    "readOnly": true,
    "resources": [{
      "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_256_GCM_SHA384",
      "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
  }
}

```

GenerateDataKey

次の例は、[GenerateDataKey](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
  "eventVersion": "1.11",

```



```

    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "keySpec": "AES_256",
      "encryptionContext": {
        "Department": "Engineering",
        "Project": "Alpha"
      }
    },
    "responseElements": null,
    "additionalEventData": {
      "keyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
    },
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [{
      "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_256_GCM_SHA384",
      "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
  }

```

```
}
```

GenerateDataKeyPair

次の例は、[GenerateDataKeyPair](#) オペレーションの AWS CloudTrail ログエントリを示しています。この例は、対称暗号化 AWS KMS keyで暗号化された RSA キーペアを生成するオペレーションを記録します。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2025-05-20T20:46:16Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKeyPair",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyPairSpec": "RSA_3072",
    "encryptionContext": {
      "Project": "Alpha"
    }
  },
  "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
},
"responseElements": null,
"additionalEventData": {
  "keyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
},
"requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
"eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
"readOnly": true,
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::KMS::Key",
```

```

      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
  },
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_256_GCM_SHA384",
    "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
  }
}

```

GenerateDataKeyPairWithoutPlaintext

次の例は、[GenerateDataKeyPairWithoutPlaintext](#) オペレーションの AWS CloudTrail ログエントリを示しています。この例は、対称暗号化 AWS KMS key で暗号化された RSA キーペアを生成するオペレーションを記録します。

```

{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2025-05-20T20:46:16Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKeyPairWithoutPlaintext",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyPairSpec": "RSA_4096",
    "encryptionContext": {
      "Index": "5"
    }
  },
  "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
}

```

```

    },
    "responseElements": null,
    "additionalEventData": {
        "keyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
    "eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}

```

GenerateDataKeyWithoutPlaintext

次の例は、[GenerateDataKeyWithoutPlaintext](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },

```

```

    "eventTime": "2025-05-20T20:46:16Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "keySpec": "AES_256",
      "encryptionContext": {
        "Project": "Alpha"
      }
    },
    "responseElements": null,
    "additionalEventData": {
      "keyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
    },
    "requestID": "d6b8e411-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "f7734272-9ec5-4c80-9f36-528ebbe35e4a",
    "readOnly": true,
    "resources": [{
      "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_256_GCM_SHA384",
      "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
  }
}

```

GenerateMac

次の例は、[GenerateMac](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
  "eventVersion": "1.08",

```

```

    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2022-12-23T19:26:54Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateMac",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "macAlgorithm": "HMAC_SHA_512",
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": null,
    "requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  }
}

```

GenerateRandom

次の例は、[GenerateRandom](#) オペレーションの AWS CloudTrail ログエントリを示しています。このオペレーションでは を使用しないため AWS KMS key、resources フィールドは空です。

```

{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",

```

```

    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:52:37Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateRandom",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": null,
  "requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
  "eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
  "readOnly": true,
  "resources": [],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}

```

GetKeyPolicy

次の例は、[GetKeyPolicy](#) オペレーションの AWS CloudTrail ログエントリを示しています。KMS キーのキーポリシーの表示についての詳細は、[キーポリシーを表示する](#) を参照してください。

```

{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:50:30Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GetKeyPolicy",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",

```

```

    "requestParameters": {
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "policyName": "default"
    },
    "responseElements": null,
    "requestID": "93746dd6-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "4aa7e4d5-d047-452a-a5a6-2cce282a7e82",
    "readOnly": true,
    "resources": [{
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "accountId": "111122223333"
    }],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  }
}

```

GetKeyRotationStatus

次の例は、[GetKeyRotationStatus](#) オペレーションの AWS CloudTrail ログエントリを示しています。KMS キーのキーマテリアルの自動ローテーションおよびオンデマンドローテーションの詳細については、「[ローテーション AWS KMS keys](#)」を参照してください。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2024-02-20T19:16:45Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GetKeyRotationStatus",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": null,

```



```

    "requestID": "12f9b7e8-49b9-4c1c-a7e3-34ac0cdf0467",
    "eventID": "3d082126-9e7d-4167-8372-a6cfcbed4be6",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
  },
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
    "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
  }
}

```

GetParametersForImport

次の例は、[GetParametersForImport](#) オペレーションの使用時に生成される AWS CloudTrail ログエントリを示しています。このオペレーションは、KMS キーにキーマテリアルをインポートするときに使用する公開キーとインポートトークンを返します。GetParametersForImport オペレーションを使用するか、AWS KMS コンソールを使用してパブリックキーとインポートトークンをダウンロードすると、同じ CloudTrail エントリが記録されます。 [???](#)

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2020-07-25T23:58:23Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GetParametersForImport",

```

```

    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "wrappingAlgorithm": "RSAES_OAEP_SHA_256",
      "wrappingKeySpec": "RSA_2048"
    },
    "responseElements": null,
    "requestID": "b5786406-e3c7-43d6-8d3c-6d5ef96e2278",
    "eventID": "4023e622-0c3e-4324-bdef-7f58193bba87",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  }

```

ImportKeyMaterial

次の例は、[ImportKeyMaterial](#) オペレーションの使用時に生成される AWS CloudTrail ログエントリを示しています。ImportKeyMaterial オペレーションを使用するか、AWS KMS コンソールを使用して[キーマテリアルを にインポート](#)すると、同じ CloudTrail エントリが記録されます AWS KMS key。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

```

{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",

```

```

        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2025-05-21T05:42:31Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ImportKeyMaterial",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "validTo": "May 21, 2025, 5:47:45 AM",
        "expirationModel": "KEY_MATERIAL_EXPIRES",
        "importType": "NEW_KEY_MATERIAL",
        "keyMaterialDescription": "ExampleKeyMaterialA"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyMaterialId": "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
    },
    "requestID": "89e10ee7-a612-414d-95a2-a128346969fd",
    "eventID": "c7abd205-a5a2-4430-bbfa-fc10f3e2d79f",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}

```


[illegible]

ListKeyRotations

次の例は、[ListKeyRotations](#) オペレーションの AWS CloudTrail ログエントリを示しています。KMS キーのキーマテリアルの自動ローテーションおよびオンデマンドローテーションの詳細については、「[ローテーション AWS KMS keys](#)」を参照してください。

```

{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2025-05-21T05:42:35Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "ListKeyRotations",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "includeKeyMaterial": "ALL_KEY_MATERIAL"
  },
  "responseElements": null,
  "requestID": "99c88d32-f2db-455e-8a9a-23855258a452",
  "eventID": "8ce0e74b-b9c7-45a2-96ef-83136d38068e",
  "readOnly": true,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
    "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
  }
}

```

PutKeyPolicy

次の例は、[PutKeyPolicy](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。キーポリシー更新する方法の詳細については、「[キーポリシーを変更する](#)」を参照してください。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-09-01T20:06:16Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "PutKeyPolicy",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "policyName": "default",
    "policy": "{\n  \"Version\" : \"2012-10-17\",\n  \"Id\" : \"key-default-1\",\n  \"Statement\" : [ {\n    \"Sid\" : \"Enable IAM User Permissions\",\n    \"Effect\" : \"Allow\",\n    \"Principal\" : {\n      \"AWS\" : \"arn:aws:iam::111122223333:root\"\n    },\n    \"Action\" : \"kms:*\",\n    \"Resource\" : \"*\"\n  } ]\n}",
    "bypassPolicyLockoutSafetyCheck": false
  },
  "responseElements": null,
  "requestID": "7bb906fa-dc21-4350-b65c-808ff0f72f55",
  "eventID": "c217db1f-903f-4a2f-8f88-9580182d6313",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
}
```

```
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}
```

ReEncrypt

次の例は、[ReEncrypt](#) オペレーションの AWS CloudTrail ログエントリを示しています。このログエントリの `resources` フィールドは AWS KMS keys、ソース KMS キーと送信先 KMS キーの 2 つをその順序で指定します。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2025-05-22T19:34:55Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "ReEncrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "sourceEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
    "sourceEncryptionContext": {
      "Project": "Alpha",
      "Department": "Engineering"
    },
    "destinationKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "destinationEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
    "destinationEncryptionContext": {
      "Level": "3A"
    }
  },
  "responseElements": null,
  "additionalEventData": {
```



```

    "destinationKeyMaterialId":
    "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
    "sourceKeyMaterialId":
    "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  },
  "requestID": "03769fd4-acf9-4b33-adf3-2ab8ca73aadf",
  "eventID": "542d9e04-0e8d-4e05-bf4b-4bdeb032e6ec",
  "readOnly": true,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_256_GCM_SHA384",
    "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
  }
}

```

ReplicateKey

次の例は、[ReplicateKey](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。ReplicateKey リクエストにより、ReplicateKey オペレーションおよび [CreateKey](#) オペレーションが実行されます。

マルチリージョンキーのレプリケーションについては、[マルチリージョンのレプリカキーを作成する](#)を参照してください。

```
{
```

```
"eventVersion": "1.08",
"userIdentity": {
  "type": "IAMUser",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::111122223333:user/Alice",
  "accountId": "111122223333",
  "accessKeyId": "EXAMPLE_KEY_ID",
  "userName": "Alice"
},
"eventTime": "2020-11-18T01:29:18Z",
"eventSource": "kms.amazonaws.com",
"eventName": "ReplicateKey",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.0.2.0",
"userAgent": "AWS Internal",
"requestParameters": {
  "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "replicaRegion": "us-west-2",
  "bypassPolicyLockoutSafetyCheck": false,
  "description": ""
},
"responseElements": {
  "replicaKeyMetadata": {
    "awsAccountId": "111122223333",
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "creationDate": "Nov 18, 2020, 1:29:18 AM",
    "enabled": false,
    "description": "",
    "keyUsage": "ENCRYPT_DECRYPT",
    "keyState": "Creating",
    "origin": "AWS_KMS",
    "keyManager": "CUSTOMER",
    "keySpec": "SYMMETRIC_DEFAULT",
    "customerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "encryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "multiRegion": true,
    "multiRegionConfiguration": {
      "multiRegionKeyType": "REPLICA",
      "primaryKey": {
```

```

        "arn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "region": "us-east-1"
    },
    "replicaKeys": [
        {
            "arn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "region": "us-west-2"
        }
    ]
},
    "replicaPolicy": "{\n  \"Version\": \"2012-10-17\", \n  \"Statement\": [{\n
    \"Effect\": \"Allow\", \n    \"Principal\": {\"AWS\": \"arn:aws:iam::123456789012:user/
Alice\"}, \n    \"Action\": \"kms:*\", \n    \"Resource\": \"*\" \n  }, {\n    \"Effect
\": \"Allow\", \n    \"Principal\": {\"AWS\": \"arn:aws:iam::012345678901:user/Bob\"}, \n
    \"Action\": \"kms:CreateGrant\", \n    \"Resource\": \"*\" \n  }, {\n    \"Effect\":
\"Allow\", \n    \"Principal\": {\"AWS\": \"arn:aws:iam::012345678901:user/Charlie\"}, \n
    \"Action\": \"kms:Encrypt\", \n    \"Resource\": \"*\" \n  }]\n}",
},
    "requestID": "abcdef68-63bc-11e4-bc2b-4198b6150d5c",
    "eventID": "fedcba44-6773-4f96-8763-1993aec9ae6a",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}

```

RetireGrant

次の例は、[RetireGrant](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。グラント廃止の詳細については、「[グラントの使用停止と取り消し](#)」を参照してください。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-09-01T19:39:33Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "RetireGrant",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": null,
  "additionalEventData": {
    "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
  },
  "requestID": "1d274d57-5697-462c-a004-f25fcc29fa26",
  "eventID": "0771bcfb-3e24-4332-9ac8-e1c06563eecf",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}
```

RevokeGrant

次の例は、[RevokeGrant](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。グラント取り消しの詳細については、「[グラントの使用停止と取り消し](#)」を参照してください。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-09-01T19:35:17Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "RevokeGrant",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
  },
  "responseElements": null,
  "requestID": "59d94c03-c5b7-428d-ae6e-f2c4b47d2917",
  "eventID": "07a23a39-6526-4ae2-b31e-d35fbe9e24ee",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}
```

RotateKey

これらの例は、 をローテーションするオペレーションの AWS CloudTrail ログエントリを示しています AWS KMS keys。KMS キーのローテーションの詳細については、[ローテーション AWS KMS keys](#) を参照してください。

次の例は、自動キーローテーションが有効化されている対称暗号化 KMS キーをローテーションするオペレーションの CloudTrail ログエントリを示しています。自動ローテーションの有効化については、「[ローテーション AWS KMS keys](#)」を参照してください。

EnableKeyRotation オペレーションを記録する CloudTrail ログエントリの例については、[EnableKeyRotation](#) を参照してください。

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2025-05-20T20:44:37Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "RotateKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "a24b3967-ddad-417f-9b22-2332b918db06",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "serviceEventDetails": {
    "rotationType": "AUTOMATIC",
    "keyOrigin": "AWS_KMS",
```

```

    "previousKeyMaterialId":
    "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
    "currentKeyMaterialId":
    "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "eventCategory": "Management"
}

```

次の例は、[RotateKeyOnDemand](#) オペレーションによって開始されたオンデマンドローテーションの CloudTrail ログエントリを示しています。対称暗号化 KMS キーをオンデマンドでローテーションする方法については、「」を参照してください[オンデマンドキーローテーションの実行](#)。

RotateKeyOnDemand オペレーションを記録する CloudTrail ログエントリの例については、[RotateKeyOnDemand](#) を参照してください。

```

{
  "eventVersion": "1.11",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2025-05-20T20:44:37Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "RotateKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "a24b3967-ddad-417f-9b22-2332b918db06",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "recipientAccountId": "111122223333",

```

```

    "serviceEventDetails": {
      "rotationType": "ON_DEMAND",
      "keyOrigin": "EXTERNAL",
      "previousKeyMaterialId":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
      "currentKeyMaterialId":
"96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068",
      "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "eventCategory": "Management"
  }
}

```

RotateKeyOnDemand

次の例は、[RotateKeyOnDemand](#) オペレーションの AWS CloudTrail ログエントリを示しています。キーがローテーションされたときに書き込まれる CloudTrail ログエントリの例については、「[RotateKey](#)」を参照してください。KMS キーのキーマテリアルのオンデマンドローテーションの詳細については、「[オンデマンドキーローテーションの実行](#)」を参照してください。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2024-02-20T17:41:57Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "RotateKeyOnDemand",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "9e1dee86-eb84-42fd-8f25-e3fc7dbb32c8",
  "eventID": "00a09fbc-20d6-4a58-9b92-7da85984ab77",

```



```

    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.2",
      "cipherSuite": "ECDHE-RSA-AES256-GCM-SHA384",
      "clientProvidedHostHeader": "kms.us-east-1.amazonaws.com"
    }
  }
}

```

ScheduleKeyDeletion

これらの例は、[ScheduleKeyDeletion](#) オペレーションの AWS CloudTrail ログエントリを示しています。

キーの削除時に書き込まれる CloudTrail ログエントリの例については、「[DeleteKey](#)」を参照してください。AWS KMS keysの削除の詳細については、「[を削除する AWS KMS key](#)」を参照してください。

次の例では、単一リージョン KMS キーに対する ScheduleKeyDeletion リクエストが記録されています。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-03-23T18:58:30Z",

```

```

    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "pendingWindowInDays": 20,
        "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "keyState": "PendingDeletion",
        "deletionDate": "Apr 12, 2021 18:58:30 PM"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "3c4226b0-1e81-48a8-a333-7fa5f3cbd118",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}

```

次の例では、レプリカキーを持つマルチリージョン KMS キーに対する ScheduleKeyDeletion リクエストが記録されています。

AWS KMS は、すべてのレプリカキーが削除されるまでマルチリージョンキーを削除しないため、responseElements フィールドでは、keyState は PendingReplicaDeletion で、deletionDate フィールドは省略されます。

```

{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",

```

```

        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-28T17:59:05Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "pendingWindowInDays": 30,
        "keyId": "mrk-1234abcd12ab34cd56ef1234567890ab"
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
        "keyState": "PendingReplicaDeletion",
        "pendingWindowInDays": 30
    },
    "requestID": "12341411-d846-42a6-a476-b1cbe3011f89",
    "eventID": "abcda5f-396d-494c-9380-0c47860df5f1",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}

```

次の例では、AWS CloudHSM [カスタムキーストアに KMS キー](#)のScheduleKeyDeletionリクエストを記録します。

```

{
    "eventVersion": "1.08",
    "userIdentity": {

```

```

        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    },
    "eventTime": "2021-10-26T23:25:25Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ScheduleKeyDeletion",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
        "pendingWindowInDays": 30
    },
    "responseElements": {
        "keyId": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
        "deletionDate": "Nov 2, 2021, 11:25:25 PM",
        "keyState": "PendingDeletion",
        "pendingWindowInDays": 30
    },
    "additionalEventData": {
        "customKeyStoreId": "cks-1234567890abcdef0",
        "clusterId": "cluster-1a23b4cdefg",
        "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]"
    },
    "requestID": "abcd9f60-2c9c-4a0b-a456-d5d998f7f321",
    "eventID": "ca01996a-01b0-4edd-bbbb-25d7b6d1a6fa",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",

```

```
"eventCategory": "Management"
}
```

Sign

これらの例は、[署名](#)オペレーションの AWS CloudTrail ログエントリを示しています。

次の例は、非対称の RSA KMS キーを使用してファイルのデジタル署名を生成する [Sign](#) オペレーションの CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-03-07T22:36:44Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Sign",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "messageType": "RAW",
    "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "signingAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
  },
  "responseElements": null,
  "requestID": "8d0b35e0-46cf-48b9-be99-bf2ebc9ab9fb",
  "eventID": "107b3cac-b125-4556-9702-12a2b9afcfff",
  "readOnly": true,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
    }
  ],
}
```

```
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}
```

SynchronizeMultiRegionKey

次の例は、[がマルチリージョンキー](#)を AWS KMS 同期するときに生成される AWS CloudTrail ログエントリを示しています。同期には、マルチリージョンプライマリキーの[共有プロパティ](#)をレプリカキーにコピーするためのクロスリージョン呼び出しが含まれます。は、マルチリージョンキーを定期的に AWS KMS 同期して、関連するすべてのマルチリージョンキーが同じキーマテリアルを持つようにします。

CloudTrail ログエントリの `resources` 要素には、AWS リージョンを含む、マルチリージョンプライマリキーのキー ARN が含まれます。関連するマルチリージョンレプリカキーとそのリージョンは、このログエントリには一覧表示されません。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2020-11-18T02:04:37Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "SynchronizeMultiRegionKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": null,
  "responseElements": {
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "12345681-de97-42e9-bed0-b02ae1abd8dc",
  "eventID": "abcdec99-2b5c-4670-9521-ddb8f031e146",
  "readOnly": false,
}
```

```

"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}

```

TagResource

次の例は、[TagResource](#) オペレーションへの呼び出しの AWS CloudTrail ログエントリを示し、タグキーが、タグ値が のDepartmentタグを追加しますIT。

キーがローテーションされたときに書き込まれる UntagResource CloudTrail ログエントリの例については、「[UntagResource](#)」を参照してください。タグ付けの詳細については AWS KMS keys、「」を参照してください[のタグ AWS KMS](#)。

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2020-07-01T21:19:25Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "TagResource",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "tags": [

```

```

        {
            "tagKey": "Department",
            "tagValue": "IT"
        }
    ],
    },
    "responseElements": null,
    "requestID": "b942584a-f77d-4787-9feb-b9c5be6e746d",
    "eventID": "0a091b9b-0df5-4cf9-b667-6f2879532b8f",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}

```

UntagResource

次の例は、タグキーが のタグを削除するための [UntagResource](#) オペレーションへの呼び出しの AWS CloudTrail ログエントリを示していますDept。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

TagResource CloudTrail ログエントリの例については、「[TagResource](#)」を参照してください。AWS KMS keysのタグ付けの詳細については、[のタグ AWS KMS](#) を参照してください。

```

{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
    }
}

```



```

    },
    "eventTime": "2020-07-01T21:19:19Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "UntagResource",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "tagKeys": [
        "Dept"
      ]
    },
    "responseElements": {
      "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "cb1d507b-6015-47f4-812b-179713af8068",
    "eventID": "0b00f4b0-036e-411d-aa75-87eb4a35a4b3",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  }
}

```

UpdateAlias

次の例は、[UpdateAlias](#) オペレーションの AWS CloudTrail ログエントリを示しています。resources 要素には、エイリアスと KMS キーリソースのフィールドが含まれます。でのエイリアスの作成については AWS KMS、「」を参照してください[エイリアスを作成する](#)。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が responseElements.keyId 値に含まれます。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2020-11-13T23:18:15Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "UpdateAlias",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "aliasName": "alias/my_alias",
    "targetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": {
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "requestID": "d9472f40-63bc-11e4-bc2b-4198b6150d5c",
  "eventID": "f72d3993-864f-48d6-8f16-e26e1ae8dff0",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:alias/my_alias"
    },
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

UpdateCustomKeyStore

次の例は、[UpdateCustomKeyStore](#)オペレーションを呼び出してカスタムキーストアのクラスター ID を更新することで生成された AWS CloudTrail ログエントリを示しています。カスタムキーストアの編集に関する詳細については、「[AWS CloudHSM キーストア設定を編集する](#)」を参照してください。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-10-21T20:17:32Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "UpdateCustomKeyStore",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "customKeyStoreId": "cks-1234567890abcdef0",
    "clusterId": "cluster-1a23b4cdefg"
  },
  "responseElements": null,
  "additionalEventData": {
    "customKeyStoreName": "ExampleKeyStore",
    "clusterId": "cluster-1a23b4cdefg"
  },
  "requestID": "abcde9e1-f1a3-4460-a423-577fb6e695c9",
  "eventID": "114b61b9-0ea6-47f5-a9d2-4f2bdd0017d5",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333"
}
```

UpdateKeyDescription

次の例は、[UpdateKeyDescription](#) オペレーションを呼び出して生成された AWS CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-09-01T19:22:40Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "UpdateKeyDescription",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "description": "New key description"
  },
  "responseElements": null,
  "requestID": "8c3c1f8b-336d-4896-b034-4eb9916bc9b3",
  "eventID": "f5f3d548-2e9e-4658-8427-9dc5b1ea791",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}
```

UpdatePrimaryRegion

次の例は、[マルチリージョンキー](#)で [UpdatePrimaryRegion](#) オペレーションを呼び出すことで生成される AWS CloudTrail ログエントリを示しています。

UpdatePrimaryRegion オペレーションでは、2 つの CloudTrail ログエントリが書き込まれます。1 つはレプリカキーに変換されるマルチリージョンのプライマリキーを持つリージョンにあり、もう 1 つはプライマリキーに変換されるマルチリージョンのレプリカキーを持つリージョンにあります。

2022 年 12 月以降に記録されたこのオペレーションの CloudTrail ログエントリには、このオペレーションがキー ARN を返さない場合でも、影響を受ける KMS キーのキー ARN が `responseElements.keyId` 値に含まれます。

次の例は、マルチリージョンキーがプライマリキーからレプリカキー (us-west-2) に変更されたリージョンの UpdatePrimaryRegion の CloudTrail ログエントリを示しています。primaryRegion フィールドは、プライマリキー (ap-northeast-1) をホストするリージョンを表示します。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2021-03-10T20:23:37Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "UpdatePrimaryRegion",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
    "primaryRegion": "ap-northeast-1"
  },
  "responseElements": {
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
}
```

```

    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "3c4226b0-1e81-48a8-a333-7fa5f3cbd118",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
  }
}

```

次の例は、マルチリージョンキーがプライマリキーからレプリカキー (ap-northeast-1) に変更されたリージョンの UpdatePrimaryRegion の CloudTrail ログエントリを示しています。このログエントリは、以前のプライマリリージョンを識別しません。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice",
    "invokedBy": "kms.amazonaws.com"
  },
  "eventTime": "2021-03-10T20:23:37Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "UpdatePrimaryRegion",
  "awsRegion": "ap-northeast-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
    "primaryRegion": "ap-northeast-1"
  },
}

```

```

    "responseElements": {
      "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "requestID": "ee408f36-ea01-422b-ac14-b0f147c68334",
    "eventID": "091e6be5-737f-43c6-8431-e3679d6d0619",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
  }
}

```

VerifyMac

次の例は、[VerifyMac](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-03-31T19:25:54Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "VerifyMac",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "macAlgorithm": "HMAC_SHA_384",
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": null,
  "requestID": "f35da560-edff-4d6e-9b40-fb306fa9ef1e",
  "eventID": "6b464487-6dea-44cd-84ad-225d7450c975",
  "readOnly": true,
  "resources": [

```

```
{
  "accountId": "111122223333",
  "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
},
{
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333",
  "eventCategory": "Management"
}
```

Verify

これらの例は、[Verify](#) オペレーションの AWS CloudTrail ログエントリを示しています。

次の例は、非対称の RSA KMS キーを使用してファイルのデジタル署名を検証する [Verify](#) オペレーションの CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2022-03-07T22:50:41Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Verify",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "signingAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256",
    "keyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "messageType": "RAW"
  },
  "responseElements": null,
  "requestID": "c73ab82a-af82-4750-ae2c-b6bb790e9c28",
  "eventID": "3b4331cd-5b7b-4de5-bf5f-82ec22f0dac0",
  "readOnly": true,
```



```
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}
```

Amazon EC2 の例1

次の例では、Amazon EC2 マネジメントコンソールで、デフォルトのボリュームキーを使用して暗号化されたボリュームを作成する IAM プリンシパルが記録されています。

次の例は、ユーザー Alice が Amazon EC2 マネジメントコンソールのデフォルトのボリュームキーを使用して作成した、暗号化されたボリュームの CloudTrail ログエントリを示しています。EC2 ログファイルレコードには、値が "vol-13439757" の volumeId フィールドが含まれます。AWS KMS レコードには、値が の encryptionContext フィールドが含まれます "aws:ebs:id": "vol-13439757"。同様に、2 つのレコード間の principalId と accountId が一致します。レコードは、暗号化されたボリュームを作成するとボリュームコンテンツの暗号化に使用されるデータキーが生成されるという事実を示しています。

```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
      },
      "eventTime": "2014-11-05T20:50:18Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "CreateVolume",
```

```

    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "size": "10",
      "zone": "us-east-1a",
      "volumeType": "gp2",
      "encrypted": true
    },
    "responseElements": {
      "volumeId": "vol-13439757",
      "size": "10",
      "zone": "us-east-1a",
      "status": "creating",
      "createTime": 1415220618876,
      "volumeType": "gp2",
      "iops": 30,
      "encrypted": true
    },
    "requestID": "1565210e-73d0-4912-854c-b15ed349e526",
    "eventID": "a3447186-135f-4b00-8424-bc41f1a93b4f",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  },
  {
    "eventVersion": "1.02",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2014-11-05T20:50:19Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "&AWS; Internal",
    "requestParameters": {
      "encryptionContext": {
        "aws:ebs:id": "vol-13439757"
      }
    },

```

```

        "numberOfBytes": 64,
        "keyId": "alias/aws/ebs"
    },
    "responseElements": null,
    "requestID": "create-123456789012-758241111-1415220618",
    "eventID": "4bd2a696-d833-48cc-b72c-05e61b608399",
    "readOnly": true,
    "resources": [
        {
            "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "accountId": "111122223333"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
]
}
```

Amazon EC2 の例 2

次の例では、Amazon EC2 インスタンスを実行する IAM プリンシパルにより、KMS キーで暗号化されたデータボリュームを作成してマウントします。このアクションは、複数の CloudTrail ログレコードを生成します。Amazon EBS と暗号化の詳細については、[「Amazon EBS 暗号化の要件」](#)を参照してください。

ボリュームが作成されると、Amazon EC2 は顧客に代わって動作し、暗号化されたデータキーを AWS KMS () から取得します `GenerateDataKeyWithoutPlaintext`。次に、データキーの復号を可能にする許可 (`CreateGrant`) を作成します。ボリュームがマウントされると、Amazon EC2 は AWS KMS を呼び出してデータキー () を復号します `Decrypt`。

Amazon EC2 インスタンスの `instanceId` である `"i-81e2f56c"` が `RunInstances` イベントに表示されます。同じインスタンス ID が、作成された許可 (`"111122223333:aws:ec2-infrastructure:i-81e2f56c"`) の `granteePrincipal` と、`Decrypt` 呼び出し (`"arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/i-81e2f56c"`) のプリンシパルである引き受けたロールを修飾します。

データボリュームを保護する KMS キーの [キー ARN](#) は、3 つの AWS KMS 呼び出しすべて (`arn:aws:kms:us-`

west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab、GenerateDataKeyWithPlaintext および CreateGrant) に表示されますDecrypt。

```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:user/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "Alice"
      },
      "eventTime": "2014-11-05T21:35:27Z",
      "eventSource": "ec2.amazonaws.com",
      "eventName": "RunInstances",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "192.0.2.0",
      "userAgent": "AWS Internal",
      "requestParameters": {
        "instancesSet": {
          "items": [
            {
              "imageId": "ami-b66ed3de",
              "minCount": 1,
              "maxCount": 1
            }
          ]
        },
        "groupSet": {
          "items": [
            {
              "groupId": "sg-98b6e0f2"
            }
          ]
        },
        "instanceType": "m3.medium",
        "blockDeviceMapping": {
          "items": [
            {
              "deviceName": "/dev/xvda",
```

```
    "ebs": {
      "volumeSize": 8,
      "deleteOnTermination": true,
      "volumeType": "gp2"
    }
  },
  {
    "deviceName": "/dev/sdb",
    "ebs": {
      "volumeSize": 8,
      "deleteOnTermination": false,
      "volumeType": "gp2",
      "encrypted": true
    }
  }
],
"monitoring": {
  "enabled": false
},
"disableApiTermination": false,
"instanceInitiatedShutdownBehavior": "stop",
"clientToken": "XdKUT141516171819",
"ebsOptimized": false
},
"responseElements": {
  "reservationId": "r-5ebc9f74",
  "ownerId": "111122223333",
  "groupSet": {
    "items": [
      {
        "groupId": "sg-98b6e0f2",
        "groupName": "launch-wizard-2"
      }
    ]
  }
},
"instancesSet": {
  "items": [
    {
      "instanceId": "i-81e2f56c",
      "imageId": "ami-b66ed3de",
      "instanceState": {
        "code": 0,
        "name": "pending"
      }
    }
  ]
}
```

```
    },
    "amiLaunchIndex": 0,
    "productCodes": {

    },
    "instanceType": "m3.medium",
    "launchTime": 1415223328000,
    "placement": {
      "availabilityZone": "us-east-1a",
      "tenancy": "default"
    },
    "monitoring": {
      "state": "disabled"
    },
    "stateReason": {
      "code": "pending",
      "message": "pending"
    },
    "architecture": "x86_64",
    "rootDeviceType": "ebs",
    "rootDeviceName": "/dev/xvda",
    "blockDeviceMapping": {

    },
    "virtualizationType": "hvm",
    "hypervisor": "xen",
    "clientToken": "XdKUT1415223327917",
    "groupSet": {
      "items": [
        {
          "groupId": "sg-98b6e0f2",
          "groupName": "launch-wizard-2"
        }
      ]
    },
    "networkInterfaceSet": {

    },
    "ebsOptimized": false
  }
]
},
"requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
```

```

    "eventID": "cd75a605-2fee-4fda-b847-9c3d330ebaae",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  },
  {
    "eventVersion": "1.02",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2014-11-05T21:35:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "constraints": {
        "encryptionContextSubset": {
          "aws:ebs:id": "vol-f67bafb2"
        }
      },
      "granteePrincipal": "111122223333:aws:ec2-infrastructure:i-81e2f56c",
      "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "responseElements": {
      "grantId": "abcde1237f76e4ba7987489ac329fbfba6ad343d6f7075dbd1ef191f0120514a"
    },
    "requestID": "41c4b4f7-8bce-4773-bf0e-5ae3bb5cbce2",
    "eventID": "c1ad79e3-0d3f-402a-b119-d5c31d7c6a6c",
    "readOnly": false,
    "resources": [
      {
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
      }
    ],
    "eventType": "AwsApiCall",

```

```

    "recipientAccountId": "111122223333"
  },
  {
    "eventVersion": "1.02",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::111122223333:user/Alice",
      "accountId": "111122223333",
      "accessKeyId": "EXAMPLE_KEY_ID",
      "userName": "Alice"
    },
    "eventTime": "2014-11-05T21:35:32Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "AWS Internal",
    "requestParameters": {
      "encryptionContext": {
        "aws:ebs:id": "vol-f67bafb2"
      },
      "numberOfBytes": 64,
      "keyId": "alias/aws/ebs"
    },
    "responseElements": null,
    "requestID": "create-111122223333-758247346-1415223332",
    "eventID": "ac3cab10-ce93-4953-9d62-0b6e5cba651d",
    "readOnly": true,
    "resources": [
      {
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
      }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  },
  {
    "eventVersion": "1.02",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "111122223333:aws:ec2-infrastructure:i-81e2f56c",

```



```

    "arn": "arn:aws:sts::111122223333:assumed-role/aws:ec2-infrastructure/
i-81e2f56c",
    "accountId": "111122223333",
    "accessKeyId": "",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2014-11-05T21:35:38Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "111122223333:aws:ec2-infrastructure",
        "arn": "arn:aws:iam::111122223333:role/aws:ec2-infrastructure",
        "accountId": "111122223333",
        "userName": "aws:ec2-infrastructure"
      }
    },
    "eventTime": "2014-11-05T21:35:47Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "requestParameters": {
      "encryptionContext": {
        "aws:ebs:id": "vol-f67bafb2"
      }
    },
    "responseElements": null,
    "requestID": "b4b27883-6533-11e4-b4d9-751f1761e9e5",
    "eventID": "edb65380-0a3e-4123-bbc8-3d1b7cff49b0",
    "readOnly": true,
    "resources": [
      {
        "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "accountId": "111122223333"
      }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  }
]

```

```
}
```

Amazon CloudWatch で KMS キーをモニタリングする

Amazon CloudWatch AWS KMS keys を使用して をモニタリングできます。Amazon CloudWatch は、 から raw データを収集し、ほぼリアルタイムの読み取り可能なメトリクス AWS KMS に処理する AWS サービスです。 [Amazon CloudWatch](#) これらのデータは、履歴情報にアクセスして、時間の経過に伴う KMS キーの使用や変更に関する理解を深められるように 2 週間記録されます。

Amazon CloudWatch を使用すると、次に示すような重要なイベントのアラートを受け取ることができます。

- KMS キーにインポートされたキーマテリアルの有効期限が近づいています。
- 削除保留中の KMS キーがまだ使用されています。
- KMS キーのキーマテリアルが自動的にローテーションされました。
- KMS キーが削除されました。

リクエストレートがクォータ値の一定の割合に達したときに警告する [Amazon CloudWatch](#) アラームを作成することもできます。詳細については、AWS セキュリティブログの [Service Quotas と Amazon CloudWatch を使用して AWS KMS API リクエストレートを管理する](#)」を参照してください。

AWS KMS メトリクスとディメンション

AWS KMS は Amazon CloudWatch メトリクスを事前定義して、重要なデータのモニタリングとアラームの作成を容易にします。AWS Management Console および Amazon CloudWatch API を使用して AWS KMS メトリクスを表示できます。

このセクションでは、各 AWS KMS メトリクスと各メトリクスのディメンションを一覧表示し、これらのメトリクスとディメンションに基づいて CloudWatch アラームを作成するための基本的なガイダンスを提供します。

Note

ディメンショングループ名:

Amazon CloudWatch コンソールでメトリクスを表示するには、[Metrics] (メトリクス) セクションでディメンショングループ名を選択します。これにより、[Metric name] (メトリクス

名) でフィルタリングできます。このトピックには、各 AWS KMS メトリクスのメトリクス名とディメンショングループ名が含まれています。

AWS Management Console および Amazon CloudWatch API を使用して AWS KMS メトリクスを表示できます。詳細については、「Amazon CloudWatch ユーザーガイド」の「[使用可能なメトリクスの表示](#)」を参照してください。

トピック

- [SuccessfulRequest](#)
- [SecondsUntilKeyMaterialExpiration](#)
- [CloudHSMKeyStoreThrottle](#)
- [ExternalKeyStoreThrottle](#)
- [XksProxyCertificateDaysToExpire](#)
- [XksProxyCredentialAge](#)
- [XksProxyErrors](#)
- [XksExternalKeyManagerStates](#)
- [XksProxyLatency](#)

SuccessfulRequest

特定の KMS キーに対する暗号化オペレーションの正常なリクエストの数。SuccessfulRequest メトリクスを使用すると、CloudWatch の AWS KMS API 使用状況にキーレベルのフィルタリングを適用できます。このメトリクスの Sum 統計は、期間中に成功したリクエストの総数を定義します。

このメトリクスを使用して、リクエストクォータの最大部分を消費するか、API 料金に最も寄与する KMS キーを特定します。SuccessfulRequest メトリクスに基づいて CloudWatch アラームを作成して、異常な AWS KMS API 使用パターンを通知することもできます。これらのアラートは、意図せずにリクエストクォータを超えたり、予想しない料金が発生する可能性がある非効率的なワークフローを特定するのに役立ちます。

のディメンション **SuccessfulRequest**

ディメンション	説明
KeyArn	各 KMS キーの値です。
Operation	各 AWS KMS API オペレーションの値。このメトリクスは、暗号化オペレーションにのみ適用されます。

[ReEncrypt](#) オペレーションの場合、SuccessfulRequestメトリクスには送信元と送信先の KMS キーの両方のディメンションが含まれます。

ディメンション	説明
SourceKeyArn	暗号文を復号した KMS キーの値。
DestinationKeyArn	データを再暗号化した KMS キーの値。
Operation	各 AWS KMS API オペレーションの値。この場合は ReEncrypt。

SecondsUntilKeyMaterialExpiration

KMS [キーでインポートされた最も早く期限切れになるキーマテリアル](#)までの残り秒数。このメトリクスは、インポートされたキーマテリアル (EXTERNAL の[キーマテリアルのオリジン](#)) と有効期限を持つ KMS キーに対してのみ有効です。

このメトリクスを使用して、最も早く期限切れになるインポートされたキーマテリアルの有効期限が切れるまでの残り時間を追跡します。その時間が定義したしきい値を下回った場合は、KMS キーを使用できるように、新しい有効期限でキーマテリアルを再インポートする必要があります。この SecondsUntilKeyMaterialExpiration メトリクスは KMS キーに固有です。このメトリクスを使用して、複数の KMS キーや将来作成する可能性のある KMS キーを監視することはできません。このメトリクスをモニタリングする CloudWatch アラームの作成に関するヘルプについては、「[インポートされたキーマテリアルの有効期限を通知する CloudWatch アラームを作成する](#)」を参照してください。

このメトリクスで最も有用な統計は Minimum で、指定された統計期間のすべてのデータポイントの最小残り時間を示します。このメトリクスの唯一の有効な単位は Seconds です。

ディメンショングループ名: [Per-Key Metrics] (キーごとのメトリクス)

SecondsUntilKeyMaterialExpiration のディメンション

ディメンション	説明、 関連 AWS
KeyId	各 KMS キーの値です。

KMS キーの[キー削除をスケジュールする](#)と、AWS KMS は KMS キーを削除する前に待機時間を強制します。待機期間を設定することで、KMS キーが不要であり、今後も使用しないことを確認できます。また、待機期間中に[暗号化操作](#)でユーザーまたはアプリケーションが KMS キーの使用を試みた場合に警告するよう、CloudWatch アラームを設定することもできます。このようなアラームから通知を受け取った場合は、KMS キーの削除をキャンセルする必要がある可能性があります。

手順については、「[削除待ち状態の KMS キーの使用を検出するアラームの作成](#)」を参照してください。

CloudHSMKeyStoreThrottle

が AWS KMS スロットリングする (と応答する ThrottlingException) 各キーストアの KMS AWS CloudHSM キーに対する暗号化オペレーションのリクエストの数。このメトリクスは AWS CloudHSM キーストアにのみ適用されます。

この CloudHSMKeyStoreThrottle メトリクスは、キーストアの AWS CloudHSM KMS キーと[暗号化オペレーション](#)のリクエストにのみ適用されます。AWS KMS [は、リクエストレートがキーストアのカスタムキーストアリクエストクォータを超えた場合に、これらのリクエストをスロットリング](#)します。[???](#) AWS CloudHSM このメトリクスには、AWS CloudHSM クラスターによるスロットリングも含まれます。

ディメンショングループ名: [Keystore Throttle Metrics] (キーストアスロットルメトリクス)

ディメンション	説明
CustomKeyStoreId	各 AWS CloudHSM キーストアの値。
KmsOperation	各 AWS KMS API オペレーションの値。このメトリクスは、キーストアの AWS CloudHSM KMS キーに対する暗号化オペレーションにのみ適用されます。
KeySpec	KMS キーの各タイプの値です。 キーストアでサポートされている KMS キーのキー仕様 は、SYMMETRIC_DEFAULT のみです。AWS CloudHSM

ExternalKeyStoreThrottle

AWS KMS スロットリングする各外部キーストアの KMS キーに対する暗号化オペレーションのリクエストの数 (と応答) ThrottlingException。このメトリクスは、[外部キーストア](#)にのみ適用されます。

ExternalKeyStoreThrottle メトリクスは、外部キーストアの KMS キーと[暗号化オペレーション](#)のリクエストにのみ適用されます。AWS KMS は、[リクエストレートが外部キーストアのカスタムキーストアリクエストクォータを超えると、これらのリクエストをスロットリングします。](#)???このメトリクスには、外部キーストアプロキシまたは外部キーマネージャーによるスロットリングは含まれていません。

このメトリクスを使用して、カスタムキーストアリクエストクォータの値を確認して調整します。このメトリクスが AWS KMS がこれらの KMS キーのリクエストを頻繁にスロットリングしていることを示している場合は、カスタムキーストアリクエストクォータ値の増加をリクエストすることを検討してください。ヘルプについては、「Service Quotas ユーザーガイド」の「[Requesting a quota increase](#)」(クォータ引き上げのリクエスト)を参照してください。

「リクエストレートが極めて高いため」リクエストが拒否されたこと、または「外部キーストアプロキシが時間内に応答しなかったために」リクエストが拒否されたことを説明するメッセージで KMSInvalidStateException エラーが頻発する場合は、外部キーマネージャーまたは外部キーストアプロキシが現在のリクエストレートに対応していないことを示している可能性があります。可能な場合は、リクエスト率を下げます。また、カスタムキーストアのリクエストクォータ値の引き下げをリクエストすることも検討してください。このクォータ値を減らすと、スロットリング (および ExternalKeyStoreThrottle メトリクス値) が増加する可能性があります。AWS KMS が外部キーストアプロキシまたは外部キーマネージャーに送信される前に、過剰なリクエストを迅速に拒

否していることを示します。クォータの削減をリクエストするには、[AWS サポート センター](#)にアクセスしてケースを作成してください。

ディメンショングループ名: [Keystore Throttle Metrics] (キーストアスロットルメトリクス)

ディメンション	説明
CustomKeyStoreId	各外部キーストアの値です。
KmsOperation	各 AWS KMS API オペレーションの値。このメトリクスは、外部キーストアの KMS キーに対する暗号化オペレーションにのみ適用されます。
KeySpec	KMS キーの各タイプの値です。外部キーストアの KMS キーでサポートされている キースペック は SYMMETRIC_DEFAULT のみです。

XksProxyCertificateDaysToExpire

[外部キーストアプロキシエンドポイント](#) (XksProxyUriEndpoint) の TLS 証明書の有効期限が切れるまでの日数です。このメトリクスは、[外部キーストア](#)にのみ適用されます。

このメトリクスを使用して、TLS 証明書の有効期限切れが近づいていることを通知する CloudWatch アラームを作成します。証明書の有効期限が切れると、AWS KMS は外部キーストアプロキシと通信できません。証明書が更新されるまで、外部キーストアの KMS キーで保護されているすべてのデータにアクセスできなくなります。

証明書アラームは、暗号化されたリソースへのアクセスを妨げる可能性のある証明書の有効期限切れを防ぎます。アラームを設定することで、組織は有効期限が切れる前に証明書を更新する時間をもつことができます。

ディメンショングループ名: [XKS Proxy Certificate Metrics] (XKS プロキシ証明書メトリクス)

ディメンション	説明
CustomKeyStoreId	各外部キーストアの値です。

ディメンション	説明
CertificateName	TLS 証明書のサブジェクト名 (CN) です。

外部キーストアと外部キーストアの KMS キーのメトリクスに基づいて CloudWatch アラームを作成できます。手順については、「[外部キーストアをモニタリングする](#)」を参照してください。

XksProxyCredentialAge

現在の外部キーストアの[プロキシ認証情報](#) (XksProxyAuthenticationCredential) が外部キーストアに関連付けられてからの日数です。このカウントは、外部キーストアの作成または更新の一環として、認証情報を入力した時点から開始されます。このメトリクスは、[外部キーストア](#)にのみ適用されます。

この値は、認証情報の有効期限を知らせるためのものです。ただし、外部キーストアプロキシで認証情報を作成した時点ではなく、認証情報を外部キーストアに関連付けた時点からカウントが開始されるため、プロキシでの認証情報の経過時間の正確なインジケータではない場合があります。

このメトリクスを使用して、外部キーストアのプロキシ認証情報をローテーションするように通知する CloudWatch アラームを作成します。

ディメンショングループ名: [Per-Keystore Metrics] (キーストアごとのメトリクス)

ディメンション	説明
CustomKeyStoreId	各外部キーストアの値です。

外部キーストアと外部キーストアの KMS キーのメトリクスに基づいて CloudWatch アラームを作成できます。手順については、「[外部キーストアをモニタリングする](#)」を参照してください。

XksProxyErrors

[外部キーストアプロキシ](#)への AWS KMS リクエストに関連する例外の数。この数には、外部キーストアプロキシが 250 ミリ秒のタイムアウトに返す例外 AWS KMS と、外部キーストアプロキシが 250 ミリ秒のタイムアウト

間隔 AWS KMS 内に応答しない場合に発生するタイムアウトエラーが含まれます。このメトリクスは、[外部キーストア](#)にのみ適用されます。

このメトリクスを使用して、外部キーストアの KMS キーのエラーレートを追跡します。最も頻発するエラーが明らかになるため、エンジニアリング作業に優先順位を付けることができます。例えば、再試行不可能なエラーの発生率が高くなっている KMS キーは、外部キーストアの設定に問題があることを示している可能性があります。外部キーストア設定を確認するには、「[外部キーストアを表示する](#)」を参照してください。外部キーストア設定を編集するには、「[外部キーストアプロパティを編集する](#)」を参照してください。

ディメンショングループ名: [XKS Proxy Error Metrics] (XKS プロキシエラーメトリクス)

ディメンション	説明
CustomKeyStoreId	各外部キーストアの値です。
KmsOperation	XKS プロキシへのリクエストを生成した各 AWS KMS API オペレーションの値。
XksOperation	各 外部キーストアプロキシ API オペレーション の値です。
KeySpec	KMS キーの各タイプの値です。外部キーストアの KMS キーでサポートされている キースペック は SYMMETRIC_DEFAULT のみです。
ErrorType	値: - 再試行可能なエラー: ネットワークエラーなど、一時的なエラーである可能性があります。 - 再試行不可能なエラー: カスタムキーストア設定または外部コンポーネントの問題を示している可能性があります。 - N/A: リクエストに成功、エラーなし
ExceptionName	値: - 例外の名前 - なし: リクエストに成功、エラーなし

外部キーストアと外部キーストアの KMS キーのメトリクスに基づいて CloudWatch アラームを作成できます。手順については、「[外部キーストアをモニタリングする](#)」を参照してください。

XksExternalKeyManagerStates

以下の各ヘルス状態 (Active、Degraded、Unavailable) における[外部キーマネージャーインスタンス](#)数のカウントです。このメトリクスの情報は、各外部キーストアに関連付けられた外部キーストアプロキシから取得されます。このメトリクスは、[外部キーストア](#)にのみ適用されます。

以下は、外部キーストアに関連付けられた外部キーマネージャーインスタンスのヘルス状態です。各外部キーストアプロキシは、外部キーマネージャーのヘルス状態を測定するために、異なるインジケータを使用する場合があります。詳細については、外部キーストアプロキシのドキュメントを参照してください。

- Active: 外部キーマネージャーは正常です。
- Degraded: 外部キーマネージャーに異常がありますが、トラフィックは処理できます。
- Unavailable: 外部キーマネージャーはトラフィックを処理できません。

このメトリクスを使用して、外部キーマネージャーインスタンスが劣化して使用できなくなったことを警告する CloudWatch アラームを作成します。各状態の外部キーマネージャーインスタンスを判断するには、外部キーストアプロキシログを参照してください。

ディメンショングループ名: [XKS External Key Manager Metrics] (XKS 外部キーマネージャーメトリクス)

ディメンション	説明
CustomKeyStoreId	各外部キーストアの値です。
XksExternalKeyManagerState	各ヘルス状態の値です。

外部キーストアと外部キーストアの KMS キーのメトリクスに基づいて CloudWatch アラームを作成できます。手順については、「[外部キーストアをモニタリングする](#)」を参照してください。

XksProxyLatency

外部キーストアプロキシが AWS KMS リクエストに応答するまでにかかるミリ秒数です。リクエストがタイムアウトした場合、記録される値は 250 ミリ秒のタイムアウト制限です。このメトリクスは、[外部キーストア](#)にのみ適用されます。

このメトリクスを使用して、外部キーストアプロキシと外部キーマネージャーのパフォーマンスを評価します。プロキシが暗号化オペレーションと復号オペレーションで頻繁にタイムアウトする場合は、外部プロキシ管理者に相談してください。

レスポンスが遅い場合は、外部キーマネージャーが現在のリクエストトラフィックを処理できないことを示している場合もあります。AWS KMS では、外部キーマネージャーが 1 秒あたり最大 1,800 件の暗号化オペレーションのリクエストを処理できることをお勧めします。外部キーマネージャーが 1 秒あたり 1,800 件のリクエストを処理できない場合は、[カスタムキーストアの KMS キーリクエストクォータ](#)の引き下げをリクエストすることを検討してください。外部キーストアの KMS キーを使用した暗号化オペレーションのリクエストは、外部キーストアプロキシまたは外部キーマネージャーによって処理され、後で拒否されるのではなく、[スロットリング例外](#)でフェイルファストします。

ディメンショングループ名: [XKS Proxy Latency Metrics] (XKS プロキシレイテンシーメトリクス)

ディメンション	説明
CustomKeyStoreId	各外部キーストアの値です。
KmsOperation	XKS プロキシへのリクエストを生成した各 AWS KMS API オペレーションの値。
XksOperation	各 外部キーストアプロキシ API オペレーション の値です。
KeySpec	KMS キーの各タイプの値です。外部キーストアの KMS キーでサポートされている キースペック は SYMMETRIC_DEFAULT のみです。

外部キーストアと外部キーストアの KMS キーのメトリクスに基づいて CloudWatch アラームを作成できます。手順については、「[外部キーストアをモニタリングする](#)」を参照してください。

インポートされたキーマテリアルの有効期限を通知する CloudWatch アラームを作成する

KMS キーのインポートされたキーマテリアルの有効期限が近づいたときに通知する CloudWatch アラームを作成できます。例えば、有効期限が 30 日未満になるとアラームが通知します。

[キーマテリアルを KMS キーにインポート](#)すると、キーマテリアルの有効期限の日時を任意で指定することができます。キーマテリアルの有効期限が切れると、キーマテリアル AWS KMS を削除し、KMS キーは使用できなくなります。KMS キーを再度使用するには、[キーマテリアルを再インポート](#)する必要があります。ただし、有効期限が切れる前にキーマテリアルを再インポートすると、その KMS キーを使用するプロセスが中断されるのを防ぐことができます。

このアラームは、インポートされたキーマテリアルの有効期限が切れた KMS キーについて AWS KMS が CloudWatch に発行する [SecondsUntilKeyMaterialExpires メトリクス](#)を使用します。各アラームはこのメトリクスを使用して、特定の KMS キーのインポートされたキーマテリアルを監視します。キーマテリアルの有効期限が近づいているすべての KMS キーに対して単一のアラームを作成したり、将来作成する可能性のある KMS キーに対してアラームを作成したりすることはできません。

要件

インポートされたキーマテリアルの有効期限を監視する CloudWatch アラームには、次のリソースが必要です。

- 有効期限が切れる、インポートされたキーマテリアルを含む KMS キー。
- Amazon SNS トピック 詳細については、「Amazon CloudWatch ユーザーガイド」の「[Amazon SNS トピックの作成](#)」を参照してください。

アラームの作成

「[静的しきい値に基づいて CloudWatch アラームを作成する](#)」の指示に従ってください。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
メトリクスの選択	[KMS] を選択し、次に [キーごとのメトリクス] を選択します。 KMS キーと SecondsUntilKeyMaterialExpires メトリクスを含む行を選択します。次に [Select metric] (メトリクスの選択) を選択します。

フィールド	値
	メトリクスリストには、インポートされたキーマテリアルが有効期限切れになっている KMS キーの <code>SecondsUntilKeyMaterialExpires</code> メトリクスのみが表示されます。アカウントとリージョンにこれらのプロパティを持つ KMS キーがない場合、このリストは空になります。
統計	最小値
期間	1 分
しきい値タイプ	静的
Whenever ...	<code>#####</code> が 1 より大きい場合は必ず

外部キーストアの CloudWatch アラームを作成する

外部キーストアメトリクスに基づいて Amazon CloudWatch アラームを作成し、メトリクス値が指定したしきい値を超えたときに通知するように設定できます。アラームは、[Amazon Simple Notification Service \(Amazon SNS\)](#) トピックまたは [Amazon EC2 Auto Scaling](#) ポリシーにメッセージを送信できます。CloudWatch アラームの詳細については、『Amazon CloudWatch ユーザーガイド』の「[Amazon CloudWatch アラームの使用](#)」を参照してください。

Amazon CloudWatch アラームを作成する前に、Amazon SNS トピックが必要です。詳細については、「Amazon CloudWatch ユーザーガイド」の「[Amazon SNS トピックの作成](#)」を参照してください。

トピック

- [証明書の有効期限切れアラームを作成する](#)
- [レスポンスタイムアウトのアラームを作成する](#)
- [再試行可能なエラーのアラームを作成する](#)
- [再試行不可能なエラーのアラームを作成する](#)

証明書の有効期限切れアラームを作成する

このアラームは、AWS KMS が CloudWatch に公開する [XksProxyCertificateDaysToExpire](#) メトリクスを使用して、外部キーストアプロキシエンドポイントに関連付けられた TLS 証明書の想定される

有効期限を記録します。アカウント内のすべての外部キーストアに対して 1 つのアラームを作成したり、今後作成する可能性のある外部キーストアに対してアラームを作成したりすることはできません。

証明書の有効期限が切れる 10 日前に警告するようアラームを設定することをお勧めしますが、ニーズに最適なしきい値を設定する必要があります。

アラームの作成

「[静的しきい値に基づいて CloudWatch アラームを作成する](#)」の指示に従ってください。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
メトリクスの選択	[KMS]、[XKS Proxy Certificate Metrics] (XKS プロキシ証明書メトリクス) の順に選択します。 モニタリングする XksProxyCertificateName の横にあるチェックボックスをオンにします。 次に [Select metric] (メトリクスの選択) を選択します。
統計	最小値
期間	5 分
しきい値タイプ	静的
Whenever ...	[XksProxyCertificateDaysToExpire] が 10 よりも Lower である場合は常に。

レスポンスタイムアウトのアラームを作成する

このアラームは、が CloudWatch AWS KMS に発行する [XksProxyLatency](#) メトリクスを使用して、外部キーストアプロキシが AWS KMS リクエストに応答するのにかかるミリ秒数を記録します。アカウント内のすべての外部キーストアに対して 1 つのアラームを作成したり、今後作成する可能性のある外部キーストアに対してアラームを作成したりすることはできません。

AWS KMS は、外部キーストアプロキシが 250 ミリ秒以内に各リクエストに応答することを期待します。外部キーストアプロキシが応答に 200 ミリ秒以上かかった場合に警告するアラームを設定することをお勧めしますが、ニーズに最適なしきい値を設定する必要があります。

アラームの作成

「[静的しきい値に基づいて CloudWatch アラームを作成する](#)」の指示に従ってください。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
メトリクス の選 択	[KMS]、[XKS Proxy Latency Metrics] (XKS プロキシレイテンシーメトリクス) の順に選択します。 モニタリングする KmsOperation の横にあるチェックボックスをオンにします。 次に [Select metric] (メトリクスの選択) を選択します。
統計	平均
期間	5 分
しきい値タイプ	静的
Whenever ...	[XksProxyLatency] が 200 よりも Greater の場合は常に。

再試行可能なエラーのアラームを作成する

このアラームは、が CloudWatch AWS KMS に発行する [XksProxyErrors](#)メトリクスを使用して、外部キーストアプロキシへの AWS KMS リクエストに関連する例外の数を記録します。アカウント内のすべての外部キーストアに対して 1 つのアラームを作成したり、今後作成する可能性のある外部キーストアに対してアラームを作成したりすることはできません。

再試行可能なエラーは信頼性の割合を低下させます。また、ネットワークエラーを示している可能性があります。1 分間に再試行可能なエラーが 5 回以上記録された場合に警告するアラームを設定することをお勧めしますが、ニーズに最適なしきい値を設定する必要があります。

「[静的しきい値に基づいて CloudWatch アラームを作成する](#)」の指示に従ってください。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
メトリクスの選択	[クエリ] タブを選択します。 [Namespace] (名前空間) として AWS/KMS を選択します。 [Metric name] (メトリクス名) に SUM(XksProxyErrors) を入力します。 [Filter by] (フィルタリング基準) に ErrorType = Retryable を入力します。 [Run] (実行) を選択します。次に [Select metric] (メトリクスの選択) を選択します。
ラベル	#####
期間	1 分
しきい値タイプ	静的
Whenever ...	[q1] が 5 よりも Greater の場合はいつでも。

再試行不可能なエラーのアラームを作成する

このアラームは、が CloudWatch AWS KMS に発行する [XksProxyErrors](#) メトリクスを使用して、外部キーストアプロキシへの AWS KMS リクエストに関連する例外の数を記録します。アカウント内のすべての外部キーストアに対して 1 つのアラームを作成したり、今後作成する可能性のある外部キーストアに対してアラームを作成したりすることはできません。

再試行不可能なエラーは、外部キーストアの設定に問題があることを示している可能性があります。1 分間に再試行不可能なエラーが 5 回以上記録された場合に警告するアラームを設定することをお勧めしますが、ニーズに最適なしきい値を設定する必要があります。

「[静的しきい値に基づいて CloudWatch アラームを作成する](#)」の指示に従ってください。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
メトリクスの選択	[クエリ] タブを選択します。 [Namespace] (名前空間) として AWS/KMS を選択します。 [Metric name] (メトリクス名) に SUM(XksProxyErrors) を入力します。 [Filter by] (フィルタリング基準) に ErrorType = Non-retryable を入力します。 [Run] (実行) を選択します。次に [Select metric] (メトリクスの選択) を選択します。
ラベル	#####
期間	1 分
しきい値タイプ	静的
Whenever ...	[q1] が 5 よりも Greater の場合はいつでも。

Amazon EventBridge を使用した KMS キーのモニタリング

Amazon EventBridge (以前の Amazon CloudWatch Events) を使用して、KMS キーのライフサイクルにおける次の重要なイベントについて警告を受け取ることができます。

- KMS キーのキーマテリアルは、自動またはオンデマンドでローテーションされました。
- KMS キーにインポートされたキーマテリアルの有効期限が切れました。
- 削除が予定されていた KMS キーが削除されました。

AWS KMS は Amazon EventBridge と統合され、KMS キーに影響する重要なイベントを通知します。各イベントは、[JSON \(JavaScript Object Notation\)](#) で表され、イベント名、イベント日時、その影響などが含まれています。これらのイベントを収集し、AWS Lambda 関数、Amazon SNS トピック、Amazon SQS キュー、Amazon Kinesis Data Streams のストリーム、組み込みターゲットなどの 1 つ以上のターゲットにルーティングするルールを確立できます。

EventBridge を他の種類のイベントで使用方法の詳細については、[「Amazon EventBridge ユーザーガイド AWS CloudTrail」](#)を参照してください。

以下のトピックでは、が AWS KMS 生成する EventBridge イベントについて説明します。

KMS CMK ローターション

AWS KMS は、対称暗号化 KMS キーでキーマテリアル[の自動ローテーションとオンデマンドローテーション](#)をサポートします。

はキーマテリアルを AWS KMS ローターションするたびに、KMS CMK Rotation イベントを EventBridge に送信します。はこのイベントをベストエフォートベースで AWS KMS 生成します。

このイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "KMS CMK Rotation",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "key-origin": "AWS_KMS",
    "rotation-type": "ON_DEMAND",
    "previous-key-material-id":
      "123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0",
    "current-key-material-id":
      "96083e4fb6dbc41d77578a213a6b6669c044dd4c143e96755396d2bf11fd6068"
  }
}
```

KMS でインポートされたキーマテリアルの有効期限

[キーマテリアルを KMS キーにインポートする](#)と、キーマテリアルの有効期限を任意で指定することができます。キーマテリアルの有効期限が切れると、はキーマテリアル AWS KMS を削除し、対応

するKMS Imported Key Material Expirationイベントを EventBridge に送信します。はこのイベントをベストエフォートベースで AWS KMS 生成します。

このイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "9da9af57-9253-4406-87cb-7cc400e43465",
  "detail-type": "KMS Imported Key Material Expiration",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ],
  "detail": {
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "key-material-id":
"123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef0"
  }
}
```

KMS CMK 削除

KMS キーの[キー削除をスケジュールする](#)と、AWS KMS は KMS キーを削除する前に待機時間を強制します。待機期間の終了後、AWS KMS は KMS キーを削除し、EventBridge へ KMS CMK Deletion イベントを送信します。AWS KMS はこの EventBridge を保証します。再試行により、同じ KMS キーを削除する複数のイベントが数秒以内に生成される場合があります。

このイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "e9ce3425-7d22-412a-a699-e7a5fc3fbc9a",
  "detail-type": "KMS CMK Deletion",
  "source": "aws.kms",
  "account": "111122223333",
  "time": "2025-05-23T03:11:54Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  ]
}
```

```
],  
  "detail": {  
    "key-id": "1234abcd-12ab-34cd-56ef-1234567890ab"  
  }  
}
```

のエイリアス AWS KMS

エイリアスは、AWS KMS keyのわかりやすい名前です。例えば、エイリアスを使用すると、1234abcd-12ab-34cd-56ef-1234567890ab の代わりに KMS キーを test-key として参照できます。

エイリアスを使用して、AWS KMS コンソール、[DescribeKey](#) オペレーション、および [Encrypt](#) や [GenerateDataKey](#) などの[暗号化オペレーション](#)で KMS キーを識別できます。エイリアスを使用すると、[AWS マネージドキー](#) の認識が容易になります。これらの KMS キーのエイリアスは、常に `aws/<service-name>` の形式になります。たとえば、Amazon DynamoDB AWS マネージドキーのエイリアスは `aws/dynamodb` です。プロジェクトやカテゴリの名前をエイリアスの前に付けるなど、プロジェクトに対して同様のエイリアス標準を設定できます。

ポリシーを編集したり、権限を管理したりすることなく、エイリアスに基づいて KMS キーへのアクセスを許可および拒否することもできます。この機能は、[属性ベースのアクセスコントロール \(ABAC\)](#) AWS KMS のサポートの一部です。詳細については、「[エイリアスを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

エイリアスの機能の強みは、エイリアスに関連付けられている KMS キーをいつでも変更できることです。エイリアスを使用すると、コードの記述と保守が容易になります。例えば、エイリアスを使用して特定の KMS キーを参照し、KMS キーを変更するとします。この場合は、単にエイリアスを別の KMS キーに関連付けます。コードを変更する必要はありません。

エイリアスを使用すると、別の AWS リージョンで同じコードを再利用することも容易になります。複数のリージョンで同じ名前のエイリアスを作成し、各エイリアスをそのリージョンの KMS キーに関連付けます。コードが各リージョンで実行されると、エイリアスは関連付けられた KMS キーをそのリージョンで参照します。例については、[アプリケーションでエイリアスを使用する方法について説明します](#)。を参照してください。

AWS KMS コンソール、[CreateAlias](#) API、または [AWS::KMS::Alias AWS CloudFormation テンプレートを使用して、KMS](#) キーのエイリアスを作成できます。

AWS KMS API は、各アカウントとリージョンのエイリアスを完全に制御します。API には、エイリアスの作成 ([CreateAlias](#))、エイリアス名とエイリアス ARN の表示 ([ListAliases](#))、エイリアスに関連付けられている KMS キーの変更 ([UpdateAlias](#))、エイリアスの削除 ([DeleteAlias](#)) のオペレーションが含まれます。

エイリアスの仕組み

AWS KMSにおけるエイリアスの仕組みについて説明します。

エイリアスは独立した AWS リソースです

エイリアスは、KMS キーのプロパティではありません。エイリアスに対して実行するアクションは、エイリアスに関連付けられた KMS キーには影響しません。KMS キーのエイリアスを作成してエイリアスを更新し、別の KMS キーに関連付けることができます。関連付けられた KMS キーに影響を与えずに、エイリアスを削除することもできます。ただし、KMS キーを削除すると、その KMS キーに関連付けられているすべてのエイリアスが削除されます。

IAM ポリシーでリソースとしてエイリアスを指定した場合、ポリシーは、関連付けられた KMS キーではなく、エイリアスを参照します。

各エイリアスには 2 つの形式があります。

エイリアスを作成するときは、エイリアス名を指定します。AWS KMS はエイリアス ARN を作成します。

- [エイリアス ARN](#) は、エイリアスを一意に識別する Amazon リソースネーム (ARN) です。

```
# Alias ARN
arn:aws:kms:us-west-2:111122223333:alias/<alias-name>
```

- [エイリアス名](#) は、アカウントとリージョンで一意です。AWS KMS API では、エイリアス名には常に というプレフィックスが付けられます alias/。このプレフィックスは AWS KMS コンソールでは省略されます。

```
# Alias name
alias/<alias-name>
```

エイリアスはシークレットではありません

エイリアスは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。エイリアス名には、機密情報や重要情報を含めないでください。

各エイリアスは、一度に 1 つの KMS キーに関連付けられる

エイリアスとその KMS キーは、同じアカウントとリージョンに存在する必要があります。

エイリアスは、同じ AWS アカウント およびリージョン内の任意の[カスターマネージドキー](#)に関連付けることができます。ただし、エイリアスを [AWS マネージドキー](#) に関連付ける許可はありません。

例えば、この [ListAliases](#) 出力では、test-key エイリアスが正確に 1 つのターゲット KMS キーに関連付けられていることが示されています。これは、TargetKeyId プロパティによって表されます。

```
{
  "AliasName": "alias/test-key",
  "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
  "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "CreationDate": 1593622000.191,
  "LastUpdatedDate": 1593622000.191
}
```

複数のエイリアスを同じ KMS キーに関連付けることができる

例えば、test-key と project-key のエイリアスを同じ KMS キーに関連付けることができます。

```
{
  "AliasName": "alias/test-key",
  "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
  "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "CreationDate": 1593622000.191,
  "LastUpdatedDate": 1593622000.191
},
{
  "AliasName": "alias/project-key",
  "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project-key",
  "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
  "CreationDate": 1516435200.399,
  "LastUpdatedDate": 1516435200.399
}
```

エイリアスは、アカウントとリージョン内で一意である必要があります

例えば、各アカウントとリージョンに test-key エイリアスを 1 つだけ持つことができます。エイリアスでは大文字と小文字が区別されますが、大文字と小文字だけが異なるエイリアスは非常にエラーが発生しやすくなります。エイリアス名は変更できません。ただし、エイリアスを削除して、目的の名前で新しいエイリアスを作成することはできます。

異なるリージョンに同じ名前のエイリアスを作成することができます

例えば、米国東部 (バージニア北部) に finance-key エイリアスを持つことができ、ヨーロッパ (フランクフルト) に finance-key エイリアスを持つことができます。各エイリアスは、そのリージョンの KMS キーに関連付けられます。コードが alias/finance-key のようなエイリアス名を参照している場合は、複数のリージョンで実行できます。各リージョンでは、異なる KMS キーが使用されます。詳細については、「[アプリケーションでエイリアスを使用する方法について説明します。](#)」を参照してください。

エイリアスに関連付けられている KMS キーを変更できる

[UpdateAlias](#) オペレーションを使用して、エイリアスを別の KMS キーに関連付けることができます。例えば、finance-key エイリアスが 1234abcd-12ab-34cd-56ef-1234567890ab KMS キーに関連付けられている場合、0987dcba-09fe-87dc-65ba-ab0987654321 KMS キーに関連付けられるようにエイリアスを更新できます。

ただし、現在の KMS キーと新しい KMS キーが同じタイプ (両方とも対称または非対称、もしくは両方とも HMAC) であり、同じ[キー使用法](#) (ENCRYPT_DECRYPT、SIGN_VERIFY、GENERATE_VERIFY_MAC のいずれか) である必要があります。この制限により、エイリアスを使用するコードのエラーが防止されます。エイリアスを別のタイプのキーに関連付ける必要があり、リスクを軽減した場合は、エイリアスを削除して再作成できます。

一部の KMS キーにはエイリアスがない

AWS KMS コンソールで KMS キーを作成するときは、新しいエイリアスを指定する必要があります。ただし、[CreateKey](#) オペレーションを使用して KMS キーを作成する場合、エイリアスは必要ありません。また、[UpdateAlias](#) オペレーションを使用してエイリアスに関連付けられている KMS キーを変更し、[DeleteAlias](#) オペレーションを使用してエイリアスを削除することもできます。そのため、KMS キーには、複数のエイリアスを持つものもあれば、エイリアスを持たないものもあります。

AWS がアカウントにエイリアスを作成する

AWS は、のアカウントにエイリアスを作成します[AWS マネージドキー](#)。これらのエイリアスには alias/aws/<service-name>、のような形式の名前 alias/aws/s3があります。

一部の AWS エイリアスには KMS キーがありません。これらの事前定義されたエイリアスは、通常、サービスの使用を開始する AWS マネージドキー ときにに関連付けられます。

エイリアスを使用して KMS キーを識別する

[エイリアス名](#)または[エイリアス ARN](#)を使用して、[暗号化オペレーション](#)、[DescribeKey](#)、[GetPublicKey](#) で KMS キーを識別できます。(KMS キーが別の AWS アカウントにある場合は、その[キー ARN](#) またはエイリアス ARN を使用する必要があります)。エイリアスは、他の AWS KMS オペレーションでは、KMS キーの有効な識別子ではありません。各 AWS KMS API オペレーションの有効な[キー識別子](#)の詳細については、AWS Key Management Service API リファレンスのKeyIdパラメータの説明を参照してください。

エイリアス名、またはエイリアス ARN を使用して、[IAM ポリシーの KMS キーを識別する](#)ことはできません。エイリアスに基づいて KMS キーへのアクセスを制御するには、[kms:RequestAlias](#) または [kms:ResourceAliases](#) 条件キーを使用します。詳細については、「[の ABAC AWS KMS](#)」を参照してください。

エイリアスへのアクセスの制御

エイリアスを作成または変更すると、エイリアスとそのエイリアスに関連付けられた KMS キーが影響を受けます。このため、エイリアスを管理するプリンシパルには、エイリアスおよび影響を受けるすべての KMS キーに対してエイリアスオペレーションを呼び出す許可が必要です。これらの許可は、[キーポリシー](#)、[IAM ポリシー](#)、[権限](#)を使用することで付与できます。

Note

タグとエイリアスを管理する許可をプリンシパルに付与する場合は注意が必要です。タグまたはエイリアスを変更すると、カスタマーマネージドキーに対するアクセス許可が許可または拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[エイリアスを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

すべての AWS KMS オペレーションへのアクセスの制御については、「」を参照してください。[アクセス許可に関するリファレンス](#)。

エイリアスを作成および管理するための権限は、次のように機能します。

kms:CreateAlias

エイリアスを作成するには、プリンシパルに、エイリアスおよび関連付けられた KMS キーの両方に対する次の許可が必要です。

- `kms:CreateAlias` エイリアスの場合。エイリアスの作成を許可されたプリンシパルにアタッチされた IAM ポリシーでこのアクセス許可を提供します。

次のポリシーステートメントの例では、Resource エlement の特定のエイリアスを指定します。ただし、複数のエイリアス ARN を一覧表示したり、「test*」などのエイリアスパターンを指定したりできます。Resource 値を "*" に指定すると、プリンシパルがアカウントとリージョンで任意のエイリアスを作成できるようになります。エイリアスを作成する権限は、アカウントおよびリージョン内のすべてのリソースに対する `kms:Create*` 権限に含めることもできます。

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms>DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- KMS キーの `kms:CreateAlias` このアクセス許可は、キーポリシーまたはキーポリシーから委任された IAM ポリシーで指定する必要があります。

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:CreateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

条件キーを使用して、エイリアスと関連付けることができる KMS キーを制限できます。例えば、[kms:KeySpec](#) 条件キーを使用して、プリンシパルが非対称 KMS キーに対してのみエイリアスを作成できるようにします。KMS キーリソースに対する `kms:CreateAlias` 許可の制限に使用できる条件キーの詳細なリストについては、[AWS KMS アクセス許可](#) を参照してください。

kms:ListAliases

アカウントとリージョンのエイリアスを一覧表示するには、プリンシパルに IAM ポリシーの `kms:ListAliases` アクセス権限が必要です。このポリシーは特定の KMS キーまたはエイリアスリソースに関連付けられていないため、ポリシー内のリソース要素の値が "*" である必要があります。

例えば、次の IAM ポリシーステートメントでは、アカウントとリージョン内のすべての KMS キーとエイリアスを一覧表示する許可をプリンシパルに付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

kms:UpdateAlias

エイリアスに関連付けられている KMS キーを変更するには、プリンシパルにエイリアス用、現在の KMS キー用、新規の KMS キー用の 3 つの許可要素が必要です。

例えば、`test-key` エイリアスをキー ID `1234abcd-12ab-34cd-56ef-1234567890ab` の KMS キーから、キー ID `0987dcba-09fe-87dc-65ba-ab0987654321` の KMS キーに変更するとします。その場合は、このセクションの例に似たポリシーステートメントを含めます。

- `kms:UpdateAlias` エイリアスの場合。このアクセス権限は、プリンシパルにアタッチされている IAM ポリシーで提供します。次の IAM ポリシーは、特定のエイリアスを指定します。ただし、複数のエイリアス ARN を一覧表示したり、`"test*"` などのエイリアスパターンを指定したりできます。`"*"` の `Resource` 値を指定することで、プリンシパルがアカウントとリージョンで任意のエイリアスを更新できるようにもなります。

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:UpdateAlias",
    "kms:ListAliases",
    "kms:ListKeys"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- 現在エイリアスに関連付けられている KMS キーの kms:UpdateAlias。このアクセス許可は、キーポリシーまたはキーポリシーから委任された IAM ポリシーで指定する必要があります。

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:UpdateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

- kms:UpdateAliasオペレーションでエイリアスに関連付けられている KMS キーの。このアクセス許可は、キーポリシーまたはキーポリシーから委任された IAM ポリシーで指定する必要があります。

```
{
  "Sid": "Key policy for 0987dcba-09fe-87dc-65ba-ab0987654321",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"},
  "Action": [
    "kms:UpdateAlias",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

条件キーを使用して、UpdateAlias オペレーションの KMS キーの一方または両方を制限できます。例えば、[kms:ResourceAliases](#) 条件キーを使用して、ターゲット KMS キーに特定のエイリアスが既に存在する場合にのみ、プリンシパルがエイリアスを更新できるようにします。KMS キーリソースに対する kms:UpdateAlias 許可の制限に使用できる条件キーの詳細なリストについては、[AWS KMS アクセス許可](#) を参照してください。

kms:DeleteAlias

エイリアスを削除するには、プリンシパルにエイリアスおよび関連付けられた KMS キーに対する許可が必要です。

プリンシパルにリソースを削除する権限を与えるときは、いつものように注意する必要があります。ただし、エイリアスを削除しても関連付けられた KMS キーには影響しません。エイリアスに依存するアプリケーションでエラーが発生する可能性があります。エイリアスを誤って削除した場合は、エイリアスを再作成することができます。

- kms:DeleteAlias エイリアスの場合。エイリアスの削除を許可されているプリンシパルにアタッチされた IAM ポリシーでこのアクセス許可を提供します。

次のポリシーステートメントの例は、Resource エレメントのエイリアスを指定します。ただし、複数のエイリアス ARN を一覧表示したり、"test*" などのエイリアスパターンを指定したりできます。"*" の Resource 値を指定して、プリンシパルにアカウントとリージョンで任意のエイリアスを削除させることもできます。

```
{
  "Sid": "IAMPolicyForAnAlias",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms:DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/test-key"
}
```

- 関連付けられた KMS キーの kms:DeleteAlias。このアクセス許可は、キーポリシーまたはキーポリシーから委任された IAM ポリシーで指定する必要があります。

```
{
  "Sid": "Key policy for 1234abcd-12ab-34cd-56ef-1234567890ab",
  "Effect": "Allow",
```

```
"Principal": {
  "AWS": "arn:aws:iam::111122223333:user/KMSAdminUser"
},
"Action": [
  "kms:CreateAlias",
  "kms:UpdateAlias",
  "kms>DeleteAlias",
  "kms:DescribeKey"
],
"Resource": "*"
}
```

エイリアスのアクセス許可を制限する

リソースが KMS キーの場合、条件キーを使用してエイリアスのアクセス許可を制限できます。例えば、次の IAM ポリシーでは、特定のアカウントとリージョンの KMS キーに対するエイリアスオペレーションを許可します。ただし、[kms:KeyOrigin](#) 条件キーを使用して、キーマテリアルを含む KMS キーへのアクセス許可をさらに制限します AWS KMS。

KMS キーリソースに対するエイリアスのアクセス許可の制限に使用できる条件キーの詳細なリストについては、[AWS KMS アクセス許可](#) を参照してください。

```
{
  "Sid": "IAMPolicyKeyPermissions",
  "Effect": "Allow",
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms>DeleteAlias"
  ],
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "AWS_KMS"
    }
  }
}
```

リソースがエイリアスのポリシーステートメントでは、条件キーを使用できません。プリンシパルが管理できるエイリアスを制限するには、エイリアスへのアクセスを制御する IAM ポリシーステートメントの Resource 要素の値を使用します。たとえば、次のポリシーステートメントでは、エイ

リソースが 始まらない限り、プリンシパルは AWS アカウント および リージョンでエイリアスを作成、更新、または削除できますRestricted。

```
{
  "Sid": "IAMPolicyForAnAliasAllow",
  "Effect": "Allow",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms>DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/*"
},
{
  "Sid": "IAMPolicyForAnAliasDeny",
  "Effect": "Deny",
  "Action": [
    "kms:CreateAlias",
    "kms:UpdateAlias",
    "kms>DeleteAlias"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:alias/Restricted*"
}
```

エイリアスを作成する

エイリアスは、AWS KMS コンソールまたは AWS KMS API オペレーションを使用して作成できます。

エイリアスは 1 ~ 256 文字の文字列である必要があります。エイリアス名に使用できるのは、英数字、スラッシュ (/)、アンダースコア (_)、およびダッシュ (-) のみです。[カスタマーマネージドキー](#)のエイリアス名を alias/aws/ で始めることはできません。alias/aws/ プレフィックスは [AWS マネージドキー](#) のために予約されます。

新規の KMS キーまたは既存の KMS キーのエイリアスを作成できます。エイリアスを追加すると、特定の KMS キーをプロジェクトまたはアプリケーションで使用できます。

AWS CloudFormation テンプレートを使用して、KMS キーのエイリアスを作成することもできます。詳細については、「AWS CloudFormation ユーザーガイド」の「[AWS::KMS::Alias](#)」を参照してください。

AWS KMS コンソールの使用

AWS KMS コンソールで [KMS キーを作成する](#) ときは、新しい KMS キーのエイリアスを作成する必要があります。既存の KMS キーのエイリアスを作成するには、KMS キーの詳細ページにある [Aliases] (エイリアス) タブをクリックします。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。AWS マネージドキー または のエイリアスを管理することはできません AWS 所有のキー。
4. テーブルで、KMS キーのキー ID またはエイリアスを選択します。次に、KMS キーの詳細ページで、[Aliases] (エイリアス) タブをクリックします。

KMS キーに複数のエイリアスがある場合は、テーブルの [Aliases] (エイリアス) 列に、1 つのエイリアスと (+n 個以上) などのエイリアスの概要が表示されます。エイリアスの概要を選択すると、KMS キーの詳細ページの [Aliases] (エイリアス) タブを直接表示します。

5. [Aliases] (エイリアス) タブで、[Create alias] (エイリアスの作成) を選択します。エイリアス名を入力し、[Create alias] (エイリアスの作成) を選択します。

Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

Note

alias/ プレフィックスを追加しないでください。コンソールが自動的に追加します。alias/ExampleAlias を入力すると、実際のエイリアス名は alias/alias/ExampleAlias になります。

AWS KMS API の使用

エイリアスを作成するには、[CreateAlias](#) オペレーションを使用します。コンソールで KMS キーを作成するプロセスとは異なり、[CreateKey](#) オペレーションでは、新規 KMS キーのエイリアスは作成されません。

Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

CreateAlias オペレーションを使用して、エイリアスを持たない新規 KMS キーのエイリアスを作成できます。CreateAlias オペレーションを使用して、既存の KMS キーにエイリアスを追加したり、誤って削除されたエイリアスを再作成したりできます。

AWS KMS API オペレーションでは、エイリアス名は `alias/` で始まり、`alias/` の後に `ExampleAlias` などの名前を付ける必要があります。エイリアスはアカウントとリージョンで一意であることが必要です。既に使用されているエイリアス名を検索するには、[ListAliases](#) オペレーションを使用します。エイリアス名では、大文字と小文字が区別されます。

TargetKeyId は、同じ AWS リージョン内の任意の[カスターマネージドキー](#)です。KMS キーを識別するには、その[キー ID](#) または [キー ARN](#) を使用します。別のエイリアスを使用することはできません。

次の例では、`example-key` エイリアスを作成し、指定した KMS キーに関連付けます。これらの例では、AWS Command Line Interface () を使用します。AWS CLI。複数のプログラミング言語の例については、「[AWS SDK または CLI CreateAlias で使用する](#)」を参照してください。

```
$ aws kms create-alias \  
  --alias-name alias/example-key \  
  --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

CreateAlias は出力を返しません。新しいエイリアスを確認するには、ListAliases オペレーションを使用します。詳細については、「[AWS KMS API の使用](#)」を参照してください。

KMS キーのエイリアス名とエイリアス ARN を見つける

エイリアスを使用すると、AWS KMS コンソールで KMS キーを簡単に認識できます。KMS キーのエイリアスは、AWS KMS コンソールまたは [ListAliases](#) オペレーションを使用して表示できます。KMS キーのプロパティを返す [DescribeKey](#) オペレーションには、エイリアスは含まれません。

次の手順は、AWS KMS コンソールと AWS KMS API を使用して KMS キーに関連付けられたエイリアスを表示および識別する方法を示しています。AWS KMS API の例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用できます。

AWS KMS コンソールの使用

AWS KMS コンソールには、KMS キーに関連付けられたエイリアスが表示されます。

1. <https://console.aws.amazon.com/kms> で AWS KMS コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ユーザーが作成および管理するアカウント内のキーを表示するには、ナビゲーションペインで [Customer managed keys] (カスタマーマネージドキー) を選択します。AWS が作成および管理しているアカウントのキーを表示するには、ナビゲーションペインでマネージドAWS キーを選択します。
4. エイリアス列には、各 KMS キーのエイリアスが表示されます。KMS キーにエイリアスがない場合は、エイリアス列にダッシュ (-) が表示されます。

KMS キーに複数のエイリアスがある場合は、エイリアス列に、(+n 個以上)などのエイリアスの概要も表示されます。例えば、次の KMS キーには 2 つのエイリアスがあり、そのうちの 1 つは key-test です。

KMS キーのすべてのエイリアスのエイリアス名とエイリアス ARN を検索するには、[Aliases] (エイリアス) タブを使用します。

- [Aliases] (エイリアス) タブに直接移動するには、[Aliases] (エイリアス) 列で、エイリアスの概要 (+n個以上) を選択します。エイリアスの概要は、KMS キーに複数のエイリアスがある場合にのみ表示されます。
- または、KMS キーのエイリアスまたはキー ID を選択し (KMS キーの詳細ページが開きます)、[Aliases] (エイリアス) タブを選択します。これらのタブは、[General configuration] (一般設定) セクションにあります。

Customer managed keys (16)				Key actions ▼	Create key
Filter keys by aliases, key ID, or key type				< 1 2 >	⚙
☐	Aliases ▼	Key ID ▼	Status		
☐	key-test (+1 more)	1234abcd-12ab-34cd-56ef-1234567890ab	Enabled		
☐	-	1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d	Enabled		

5. [Aliases] (エイリアス) タブには、KMS キーのすべてのエイリアスのエイリアス名とエイリアス ARN が表示されます。このタブで、KMS キーのエイリアスを作成または削除することもできます。

Key policy

Cryptographic configuration

Key material

Tags

Public key

Aliases

Aliases

Info

Delete

Create new alias

Q

Filter by Alias name

< 1 >

☐ Alias name	Alias ARN
☐ key-test	arn:aws:kms:us-east-1:111122223333:alias/key-test
☐ project-key	arn:aws:kms:us-east-1:111122223333:alias/project-key

AWS マネージドキー

この例のAWS マネージドキーページに示すように AWS マネージドキー、エイリアスを使用して を認識できます。AWS マネージドキー のエイリアスでは、形式は常に `aws/<service-name>` です。例えば、Amazon DynamoDB AWS マネージドキー の のエイリアスは `aws/dynamodb` です。

AWS managed keys (9)	
Filter keys by alias or key ID	
Alias	▲
aws/dynamodb	
aws/ebs	
aws/lightsail	
aws/rds	
aws/s3	
aws/secretsmanager	
aws/ssm	
aws/workmail	
aws/xray	

AWS KMS API の使用

[ListAliases](#) オペレーションは、アカウントおよびリージョン内のエイリアスのエイリアス名とエイリアス ARN を返します。出力には、カスタマーマネージドキーの AWS マネージドキー およびのエイリアスが含まれます。AWS マネージドキー のエイリアスは、aws/<service-name>の形式になります (例: aws/dynamodb)。

レスポンスには、TargetKeyId フィールドがないエイリアスが含まれている場合もあります。これらは、AWS によって作成されたが、まだ KMS キーに関連付けられていない事前定義されたエイリアスです。

```
$ aws kms list-aliases
{
  "Aliases": [
    {
      "AliasName": "alias/access-key",
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
      "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
      "CreationDate": 1516435200.399,
      "LastUpdatedDate": 1516435200.399
    },
    {
      "AliasName": "alias/ECC-P521-Sign",
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ECC-P521-Sign",
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
```

```

        "CreationDate": 1693622000.704,
        "LastUpdatedDate": 1693622000.704
    },
    {
        "AliasName": "alias/ImportedKey",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/ImportedKey",
        "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
        "CreationDate": 1493622000.704,
        "LastUpdatedDate": 1521097200.235
    },
    {
        "AliasName": "alias/finance-project",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/finance-project",
        "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
        "CreationDate": 1604958290.014,
        "LastUpdatedDate": 1604958290.014
    },
    {
        "AliasName": "alias/aws/dynamodb",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
        "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef",
        "CreationDate": 1521097200.454,
        "LastUpdatedDate": 1521097200.454
    },
    {
        "AliasName": "alias/aws/ebs",
        "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
        "TargetKeyId": "abcd1234-09fe-ef90-09fe-ab0987654321",
        "CreationDate": 1466518990.200,
        "LastUpdatedDate": 1466518990.200
    }
]
}

```

特定の KMS キーに関連付けられているすべてのエイリアスを取得するには、ListAliases オペレーションのオプションの KeyId パラメータを使用します。KeyId パラメータは、KMS キーの [キー ID](#) または [キー ARN](#) を受け取ります。

この例では、0987dcba-09fe-87dc-65ba-ab0987654321 KMS キーに関連付けられているすべてのエイリアスを取得します。

```

$ aws kms list-aliases --key-id 0987dcba-09fe-87dc-65ba-ab0987654321
{

```

```

"Aliases": [
  {
    "AliasName": "alias/access-key",
    "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
    "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "CreationDate": "2018-01-20T15:23:10.194000-07:00",
    "LastUpdatedDate": "2018-01-20T15:23:10.194000-07:00"
  },
  {
    "AliasName": "alias/finance-project",
    "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/finance-project",
    "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "CreationDate": 1604958290.014,
    "LastUpdatedDate": 1604958290.014
  }
]
}

```

KeyId パラメータはワイルドカード文字を使用しませんが、プログラミング言語の機能を使用して応答をフィルタリングできます。

たとえば、次の AWS CLI コマンドは のエイリアスのみを取得します AWS マネージドキー。

```
$ aws kms list-aliases --query 'Aliases[?starts_with(AliasName, `alias/aws/`)]'
```

次のコマンドは、 access-key エイリアスのみを取得します。エイリアス名では、大文字と小文字が区別されます。

```

$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/access-key`]'
[
  {
    "AliasName": "alias/access-key",
    "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/access-key",
    "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "CreationDate": "2018-01-20T15:23:10.194000-07:00",
    "LastUpdatedDate": "2018-01-20T15:23:10.194000-07:00"
  }
]

```

エイリアスを更新する

エイリアスは独立したリソースであるため、エイリアスに関連付けられている KMS キーを変更することができます。例えば、test-key エイリアスがある KMS キーに関連付けられている場合、[UpdateAlias](#) オペレーションを使用して、エイリアスを別の KMS キーに関連付けることができます。これは、キーマテリアルを変更せずに [KMS キーを手動でローテーションする](#) 方法の 1 つです。KMS キーを更新し、新しいリソースに特定の KMS キーを使用していたアプリケーションで、別の KMS キーを使用することもできます。

AWS KMS コンソールでエイリアスを更新することはできません。また、UpdateAlias (または他のオペレーション) を使用してエイリアス名を変更することはできません。エイリアス名を変更するには、現在のエイリアスを削除してから KMS キーの新しいエイリアスを作成します。

エイリアスを更新するときは、現在の KMS キーと新しい KMS キーが同じタイプ (両方とも対称または非対称、もしくは HMAC) である必要があります。これらのキーの用途も同じである必要があります (ENCRYPT_DECRYPT、SIGN_VERIFY、または GENERATE_VERIFY_MAC)。この制限により、エイリアスを使用するコードの暗号化エラーが防止されます。

次の例では、最初に [ListAliases](#) オペレーションを使用して、test-key エイリアスが現在、KMS キー 1234abcd-12ab-34cd-56ef-1234567890ab に関連付けられていることを示します。

```
$ aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
{
  "Aliases": [
    {
      "AliasName": "alias/test-key",
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "CreationDate": 1593622000.191,
      "LastUpdatedDate": 1593622000.191
    }
  ]
}
```

次に、UpdateAlias オペレーションを使用して、test-key エイリアスに関連付けられている KMS キーを、KMS キー 0987dcba-09fe-87dc-65ba-ab0987654321 に変更します。現在関連付けられている KMS キーを指定する必要はありません。新しい (「ターゲット」) KMS キーのみを指定します。エイリアス名では、大文字と小文字が区別されます。

```
$ aws kms update-alias --alias-name 'alias/test-key' --target-key-id
0987dcba-09fe-87dc-65ba-ab0987654321
```

エイリアスが現在、ターゲット KMS キーに関連付けられていることを確認するには、再度、ListAliases オペレーションを使用します。この AWS CLI コマンドは、test-key --query パラメータを使用してエイリアスのみを取得します。TargetKeyId および LastUpdatedDate フィールドが更新されます。

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/test-key`]'
[
  {
    "AliasName": "alias/test-key",
    "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/test-key",
    "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321",
    "CreationDate": 1593622000.191,
    "LastUpdatedDate": 1604958290.154
  }
]
```

エイリアスを削除する

エイリアスは、AWS KMS コンソールで、または [DeleteAlias](#) オペレーションを使用して削除できます。エイリアスを削除する前に、そのエイリアスが使用されていないことを確認してください。エイリアスを削除しても関連付けられている KMS キーには影響しませんが、そのエイリアスを使用するアプリケーションに問題が発生する可能性があります。エイリアスを誤って削除した場合は、同じ名前の新しいエイリアスを作成し、同じまたは別の KMS キーに関連付けることができます。

KMS キーを削除すると、その KMS キーに関連付けられているすべてのエイリアスが削除されます。

AWS KMS コンソールの使用

AWS KMS コンソールでエイリアスを削除するには、KMS キーの詳細ページのエイリアスタブを使用します。KMS キーの複数のエイリアスを一度に削除できます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。

3. ナビゲーションペインで、[カスターマネージドキー] を選択します。AWS マネージドキー または のエイリアスを管理することはできません AWS 所有のキー。
4. テーブルで、KMS キーのキー ID またはエイリアスを選択します。次に、KMS キーの詳細ページで、[Aliases] (エイリアス) タブをクリックします。

KMS キーに複数のエイリアスがある場合は、テーブルの [Aliases] (エイリアス) 列に、1 つのエイリアスと (+n 個以上) などのエイリアスの概要が表示されます。エイリアスの概要を選択すると、KMS キーの詳細ページの [Aliases] (エイリアス) タブを直接表示します。

5. [Aliases] (エイリアス) タブで、削除するエイリアスの横にあるチェックボックスをオンにします。その後、[削除] をクリックします。

AWS KMS API の使用

エイリアスを削除するには、[DeleteAlias](#) オペレーションを使用します。このオペレーションでは、エイリアスは一度に 1 つずつ削除されます。エイリアス名では、大文字と小文字が区別されます。また、エイリアス名の先頭には必ず alias/ プレフィックスが付いています。

例えば、次のコマンドは、test-key エイリアスを削除します。このコマンドは出力を返しません。

```
$ aws kms delete-alias --alias-name alias/test-key
```

エイリアスが削除されていることを確認するには、[ListAliases](#) オペレーションを使用します。次のコマンドは、の test-key --query パラメータ AWS CLI を使用してエイリアスのみを取得します。応答の空の括弧は、ListAliases 応答に test-key エイリアスが含まれていないことを示します。括弧を削除するには、--output text パラメータと値を使用します。

```
$ aws kms list-aliases --query 'Aliases[?AliasName==`alias/test-key`]'
[]
```

エイリアスを使用して KMS キーへのアクセスを制御する

KMS キーに関連付けられたエイリアスに基づいて、KMS キーへのアクセスを制御できます。これを行うには、[kms:RequestAlias](#) および [kms:ResourceAliases](#) 条件キーを使用します。この機能は、[属性ベースのアクセスコントロール](#) (ABAC) AWS KMS のサポートの一部です。

kms:RequestAlias 条件キーは、リクエスト内のエイリアスに基づいて、KMS キーへのアクセスを許可または拒否します。kms:ResourceAliases 条件キーは、KMS キーに関連付けられたエイリアスに基づいて、KMS キーへのアクセスを許可または拒否します。

これらの機能では、ポリシーステートメントの resource 要素のエイリアスを使用して KMS キーを識別することはできません。エイリアスが resource 要素の値の場合、ポリシーは関連付けられている KMS キーではなく、エイリアスリソースに適用されます。

Note

タグとエイリアスの変更が KMS キーの認可に影響を及ぼすまでに最長 5 分かかることがあります。最近の変更は、認可に影響を与える前に API オペレーションで表示される場合があります。

エイリアスを使用して KMS キーへのアクセスを制御する際は、次の点を考慮してください。

- エイリアスを使用して、[最小特権アクセス](#)のベストプラクティスを強化します。IAM プリンシパルでは、使用または管理する必要がある KMS キーのみに対して、必要なアクセス許可のみを付与します。例えば、エイリアスを使用してプロジェクトに使用される KMS キーを識別します。次に、プロジェクトエイリアスを持つ KMS キーのみを使用する許可をプロジェクトチームに付与します。
- プリンシパルにエイリアスを追加、編集、削除できる
kms:CreateAlias、kms:UpdateAlias、kms>DeleteAlias 許可を付与する際は注意してください。エイリアスを使用して KMS キーへのアクセスを制御する際、タグを変更することで、使用許可のない KMS キーに対する使用許可をプリンシパルに付与してしまう可能性があります。他のプリンシパルがジョブを実行するために必要な KMS キーへのアクセスを拒否することもできます。
- エイリアスを管理し、必要に応じてアクセス許可を調整するアクセス許可を現在持っていない AWS アカウントのプリンシパルを確認します。キーポリシーを変更したり、権限を作成したりする許可のないキー管理者も、エイリアスを管理する許可があれば、KMS キーへのアクセスを制御できます。

例えば、コンソールの[キー管理者のデフォルトキーポリシー](#)に

は、kms:CreateAlias、kms>DeleteAlias、kms:UpdateAlias アクセス許可があります。IAM ポリシーでは、AWS アカウントのすべての KMS キーに対するエイリアスのアクセス許可を付与する可能性があります。例えば、[AWSKeyManagementServicePowerUser](#) マネージドポ

リシーでは、プリンシパルはすべての KMS キーのエイリアスを作成、削除、一覧表示できますが、更新はできません。

- エイリアスに依存するポリシーを設定する前に、の KMS キーのエイリアスを確認してください AWS アカウント。含めるエイリアスにのみポリシーが適用されることを確認します。[CloudTrail ログ](#)および [CloudWatch アラーム](#)を使用して、KMS キーへのアクセスに影響する可能性のあるエイリアス変更をアラートさせます。[ListAliases](#) レスポンスには、各エイリアスの作成日と最終更新日が含まれます。
- エイリアスポリシー条件はパターンマッチングを使用します。エイリアスの特定のインスタンスには関連付けられません。エイリアススペースの条件キーを使用するポリシーは、パターンに一致するすべての新規および既存のエイリアスに影響します。ポリシー条件に一致するエイリアスを削除して再作成すると、古いエイリアスの場合と同様に、新しいエイリアスに条件が適用されます。

kms:RequestAlias 条件キーは、オペレーションリクエストで明示的に指定されたエイリアスに依存します。kms:ResourceAliases 条件キーは、リクエストに表示されない場合でも、KMS キーに関連付けられているエイリアスに依存します。

kms:RequestAlias

リクエスト内の KMS キーを識別するエイリアスに基づいて、KMS キーへのアクセスを許可または拒否します。[キーポリシー](#)または IAM ポリシーで、[kms:RequestAlias](#) 条件キーを使用できます。これは、エイリアスを使用してリクエスト内の KMS キーを識別するオペレーション、つまり [暗号化オペレーション](#)、[DescribeKey](#)、[GetPublicKey](#) に適用されます。[CreateAlias](#) または [DeleteAlias](#) のようなエイリアスオペレーションには無効です。

条件キーで、[エイリアス名](#)またはエイリアス名パターンを指定します。[エイリアス ARN](#) を指定することはできません。

例えば、次のポリシーステートメントでは、プリンシパルが KMS キーに対して、指定されたオペレーションを使用できるようにします。アクセス許可は、KMS キーを識別する alpha を含むエイリアスをリクエストが使用する場合にのみ有効です。

```
{
  "Sid": "Key policy using a request alias condition",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/alpha-developer"
  },
  "Action": [
    "kms:Decrypt",
```

```
"kms:GenerateDataKey*",
"kms:DescribeKey"
],
"Resource": "*",
"Condition": {
  "StringLike": {
    "kms:RequestAlias": "alias/*alpha*"
  }
}
}
```

次の認可済みプリンシパルからのサンプルリクエストは条件を満たします。ただし、これらの値が同じ KMS キーを識別したとしても、[キー ID](#)、[キー ARN](#)、別のエイリアスが条件を満たさない可能性があります。

```
$ aws kms describe-key --key-id "arn:aws:kms:us-west-2:111122223333:alias/project-alpha"
```

kms:ResourceAliases

KMS キーに関連付けられたエイリアスに基づいて、エイリアスがリクエストで使用されていない場合でも、KMS キーへのアクセスを許可または拒否します。[kms:ResourceAliases](#) 条件キーを使用すると、`alias/test*` のようなエイリアスまたはエイリアスパターンを指定できるため、この条件キーを IAM ポリシーで使用して、同じリージョン内の複数の KMS キーへのアクセスを制御できます。これは、KMS キーを使用するすべての AWS KMS オペレーションで有効です。

たとえば、次の IAM ポリシーでは、プリンシパルが 2 つの KMS キーで指定されたオペレーションを呼び出すことができます AWS アカウント。ただし、アクセス許可は `restricted` で始まるエイリアスに関連付けられた KMS キーにのみ適用されます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AliasBasedIAMPolicy",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
```

```
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": [
    "arn:aws:kms:*:111122223333:key/*",
    "arn:aws:kms:*:444455556666:key/*"
  ],
  "Condition": {
    "ForAnyValue:StringLike": {
      "kms:ResourceAliases": "alias/restricted*"
    }
  }
}
```

kms:ResourceAliases 条件はリクエストではなく、リソースの条件です。したがって、エイリアスを指定しないリクエストは、引き続き条件を満たすことができます。

次のリクエスト例は一致するエイリアスを指定し、条件を満たしています。

```
$ aws kms enable-key-rotation --key-id "alias/restricted-project"
```

ただし、次のリクエスト例では、指定された KMS キーに restricted で始まるエイリアスがあれば、そのエイリアスがリクエストで使用されなくても、条件を満たします。

```
$ aws kms enable-key-rotation --key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
```

アプリケーションでエイリアスを使用する方法について説明します。

エイリアスを使用して、アプリケーションコードで KMS キーを表示できます。KeyId パラメータ AWS KMS [の暗号化オペレーション](#)、[DescribeKey](#)、および [GetPublicKey](#) は、エイリアス名またはエイリアス ARN を受け入れます。

例えば、次の GenerateDataKey コマンドでは、エイリアス名 (alias/finance) を使用して KMS キーを識別します。エイリアス名は、KeyId パラメータの値です。

```
$ aws kms generate-data-key --key-id alias/finance --key-spec AES_256
```

KMS キーが異なる にはある場合は AWS アカウント、これらのオペレーションでキー ARN またはエイリアス ARN を使用する必要があります。エイリアス ARN を使用する際、KMS キーのエイリアスは KMS キーを所有するアカウントで定義され、リージョンごとに異なる場合があることに注意してください。エイリアス ARN を検索する方法については、[KMS キーのエイリアス名とエイリアス ARN を見つける](#) を参照してください。

例えば、次の `GenerateDataKey` コマンドでは、発信者のアカウントに含まれていない KMS キーを使用します。ExampleAlias エイリアスは、指定したアカウントおよびリージョンの KMS キーに関連付けられます。

```
$ aws kms generate-data-key --key-id arn:aws:kms:us-west-2:444455556666:alias/ExampleAlias --key-spec AES_256
```

エイリアスの最も強力な使用法の 1 つは、アプリケーションを複数の AWS リージョンで実行する場合です。例えば、署名と検証に RSA [非対称 KMS キー](#) を使用するグローバルなアプリケーションがあるとします。

- 米国西部 (オレゴン) (s-west-2) では、arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab を使用します。
- ヨーロッパ (フランクフルト) (eu-central-1) では arn:aws:kms:eu-central-1:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321 を、
- アジアパシフィック (シンガポール) (ap-southeast-1) では、arn:aws:kms:ap-southeast-1:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d を使用します。

各リージョンで異なるバージョンのアプリケーションを作成するか、ディクショナリまたはスイッチステートメントを使用して、各リージョンに適切な KMS キーを選択できます。ただし、各リージョンで同じエイリアス名を持つエイリアスを作成する方がはるかに簡単です。エイリアス名では、大文字と小文字が区別されます。

```
aws --region us-west-2 kms create-alias \
  --alias-name alias/new-app \
  --key-id arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab

aws --region eu-central-1 kms create-alias \
  --alias-name alias/new-app \
  --key-id arn:aws:kms:eu-central-1:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321
```

```
aws --region ap-southeast-1 kms create-alias \  
    --alias-name alias/new-app \  
    --key-id arn:aws:kms:ap-  
southeast-1:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d
```

次に、コード内でエイリアスを使用します。コードが各リージョンで実行されると、エイリアスは、そのリージョンの関連付けられた KMS キーを参照します。例えば、このコードは、エイリアス名を使用して [Sign](#) オペレーションを呼び出します。

```
aws kms sign --key-id alias/new-app \  
    --message $message \  
    --message-type RAW \  
    --signing-algorithm RSASSA_PSS_SHA_384
```

ただし、エイリアスが削除または更新され、別の KMS キーに関連付けられるリスクがあります。この場合、エイリアス名を使用した署名の検証に失敗し、エイリアスの再作成または更新が必要になる場合があります。

このリスクを軽減するには、アプリケーションで使用するエイリアスを管理する権限をプリンシパルに与えることに注意する必要があります。詳細については、「[エイリアスへのアクセスの制御](#)」を参照してください。

複数の AWS リージョン([AWS Encryption SDK](#) など) のデータを暗号化するアプリケーションには、他にも複数のソリューションがあります。

AWS CloudTrail ログでエイリアスを検索する

エイリアスを使用して、AWS KMS API オペレーション `AWS KMS key` で `key` を表すことができます。これを行うと、KMS キーのエイリアスとキー ARN がイベントの AWS CloudTrail ログエントリに記録されます。エイリアスが `requestParameters` フィールドに表示されます。キー ARN が `resources` フィールドに表示されます。これは、AWS サービスがアカウント AWS マネージドキー `key` を使用している場合でも当てはまります。

例えば、次の [GenerateDataKey](#) リクエストでは、`project-key` エイリアスを使用して KMS キーを表します。

```
$ aws kms generate-data-key --key-id alias/project-key --key-spec AES_256
```

このリクエストが CloudTrail ログに記録されると、ログエントリには実際に使用された KMS キーのエイリアスとキー ARN の両方が含まれます。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "ABCDE",
    "arn": "arn:aws:iam::111122223333:role/ProjectDev",
    "accountId": "111122223333",
    "accessKeyId": "FFHIJ",
    "userName": "example-dev"
  },
  "eventTime": "2020-06-29T23:36:41Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "205.205.123.000",
  "userAgent": "aws-cli/1.18.89 Python/3.6.10
Linux/4.9.217-0.1.ac.205.84.332.metal1.x86_64 botocore/1.17.12",
  "requestParameters": {
    "keyId": "alias/project-key",
    "keySpec": "AES_256"
  },
  "responseElements": null,
  "requestID": "d93f57f5-d4c5-4bab-8139-5a1f7824a363",
  "eventID": "d63001e2-dbc6-4aae-90cb-e5370aca7125",
  "readOnly": true,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }
  ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

CloudTrail ログのログ記録 AWS KMS オペレーションの詳細については、「」を参照してください。
[を使用した AWS KMS API コールのログ記録 AWS CloudTrail。](#)

のタグ AWS KMS

タグは、AWS リソースに割り当てる (または割り当て AWS することができる) オプションのメタデータラベルです。各タグは、タグキーとタグ値で構成され、どちらも大文字と小文字が区別される文字列です。タグ値には、空の (null) 文字列を指定できます。リソースの各タグには異なるタグキーが必要ですが、同じタグを複数の AWS リソースに追加できます。各リソースには、最大 50 個のユーザーが作成したタグを含めることができます。

タグキーまたはタグ値には、機密情報や重要情報を含めないでください。タグには AWS のサービス、請求を含む多くのユーザーがアクセスできます。

では AWS KMS、KMS キーの作成時にカスターマネージドキーにタグを追加し、[削除保留中](#)でない限り、既存の KMS キーにタグを付けるかタグを解除できます。エイリアス、カスタムキーストア AWS マネージドキーAWS 所有のキー、または KMS キーを他の でタグ付けすることはできません AWS アカウント。タグはオプションですが、非常に便利です。

例えば、Alpha プロジェクトに使用するすべての KMS キーと Amazon S3 バケットに "Project"="Alpha" タグを追加できます。

```
TagKey    = "Project"
TagValue  = "Alpha"
```

形式や構文などのタグに関する一般的な情報については、の[AWS 「リソースのタグ付け」](#)を参照してくださいAmazon Web Services 全般のリファレンス。

タグは、以下のことに役立ちます。

- AWS リソースを特定して整理します。多くの AWS サービスはタグ付けをサポートしているため、異なるサービスのリソースに同じタグを割り当てて、リソースが関連していることを示すことができます。たとえば、KMS キーと Amazon Elastic Block Store (Amazon EBS) ボリュームまたは AWS Secrets Manager シークレットに同じタグを割り当てることができます。タグを使用して、オートメーションのために KMS キーを識別することもできます。
- AWS コストを追跡します。AWS リソースにタグを追加すると、は使用量とコストをタグ別に集計したコスト配分レポート AWS を生成します。この機能を使用して、プロジェクト、アプリケーション、またはコストセンターの AWS KMS コストを追跡できます。

タグを使用したコスト配分の詳細については、「AWS Billing ユーザーガイド」の「[コスト配分タグの使用](#)」を参照してください。タグキーとタグ値に適用されるルールの詳細については、「AWS Billing ユーザーガイド」の「[ユーザー定義タグの制限](#)」を参照してください。

- AWS リソースへのアクセスを制御します。タグに基づいて KMS キーへのアクセスを許可または拒否することは、[属性ベースのアクセスコントロール \(ABAC\)](#) AWS KMS のサポートの一部です。タグ AWS KMS keys に基づいてへのアクセスを制御する方法については、「」を参照してください。[タグを使用して KMS キーへのアクセスを制御する](#)。タグを使用して AWS リソースへのアクセスを制御する方法の詳細については、IAM ユーザーガイドの「[リソースタグを使用した AWS リソースへのアクセスの制御](#)」を参照してください。

AWS KMS [TagResource](#)、[UntagResource](#)、または [ListResourceTags](#) オペレーションを使用すると、は AWS CloudTrail ログにエントリを書き込みます。

トピック

- [タグへのアクセスを制御する](#)
- [KMS キーにタグを追加する](#)
- [KMS キーに関連付けられたタグを編集する](#)
- [KMS キーに関連付けられたタグを削除する](#)
- [KMS キーに関連付けられたタグを表示する](#)
- [タグを使用して KMS キーへのアクセスを制御する](#)

タグへのアクセスを制御する

AWS KMS コンソールまたは API を使用してタグを追加、表示、削除するには、プリンシパルにタグ付けのアクセス許可が必要です。これらのアクセス許可は[キーポリシー](#)で付与できます。または、IAM ポリシー ([VPC エンドポイントポリシー](#)を含む) で付与することもできますが、[キーポリシーが許可する場合に限られます](#)。[AWSKeyManagementServicePowerUser](#) マネージドポリシーは、プリンシパルがアカウントからアクセスできるすべての KMS キーで、タグ付け、タグ解除、タグの一覧表示を行うことを許可します。

タグ AWS のグローバル条件キーを使用して、これらのアクセス許可を制限することもできます。AWS KMSではこれらの条件により、[TagResource](#) および [UntagResource](#) のようなタグ付けオペレーションへのアクセスを制御できます。

Note

タグとエイリアスを管理する許可をプリンシパルに付与する場合は注意が必要です。タグまたはエイリアスを変更すると、カスターマネージドキーに対するアクセス許可が許可または拒否されます。

は拒否される可能性があります。詳細については、「[の ABAC AWS KMS](#)」および「[タグを使用して KMS キーへのアクセスを制御する](#)」を参照してください。

サンプルポリシーおよび詳細については、IAM ユーザーガイドの[タグキーに基づいたアクセス制御](#)を参照してください。

タグを作成および管理するためのアクセス許可は、次のように機能します。

kms:TagResource

プリンシパルにタグの追加または編集を許可します。KMS キーの作成中にタグを追加するには、プリンシパルが特定の KMS キーに制限されない IAM ポリシーでアクセス許可を持っている必要があります。

kms:ListResourceTags

プリンシパルが KMS キーのタグを表示できるようにします。

kms:UntagResource

プリンシパルが KMS キーからタグを削除できるようにします。

ポリシーのタグ付け許可

キーポリシーまたは IAM ポリシーでタグ付け許可を付与できます。例えば、次のキーポリシーの例では、選択したユーザーに KMS キーに対するタグ付け許可が付与されます。これにより、サンプルの管理者ロールまたはデベロッパーロールを引き受けることができるすべてのユーザーにタグを表示する許可が付与されます。

JSON

```
{
  "Version": "2012-10-17",
  "Id": "example-key-policy",
  "Statement": [
    {
      "Sid": "EnableIAMUserPermissions",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Action": "kms:*",
```

```

    "Resource": "*"
  },
  {
    "Sid": "AllowAllTaggingPermissions",
    "Effect": "Allow",
    "Principal": {"AWS": [
      "arn:aws:iam::111122223333:user/LeadAdmin",
      "arn:aws:iam::111122223333:user/SupportLead"
    ]},
    "Action": [
      "kms:TagResource",
      "kms:ListResourceTags",
      "kms:UntagResource"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowRolesViewTags",
    "Effect": "Allow",
    "Principal": {"AWS": [
      "arn:aws:iam::111122223333:role/Administrator",
      "arn:aws:iam::111122223333:role/Developer"
    ]},
    "Action": "kms:ListResourceTags",
    "Resource": "*"
  }
]
}

```

プリンシパルに複数の KMS キーに対するタグ付け許可を付与するには、IAM ポリシーを使用します。このポリシーを有効にするには、各 KMS キーのキーポリシーで、アカウントが IAM ポリシーを使用して KMS キーへのアクセスを制御することを許可する必要があります。

例えば、次の IAM ポリシーではプリンシパルが KMS キーを作成することを許可します。指定したアカウントのすべての KMS キーでタグを作成および管理することもできます。この組み合わせにより、プリンシパルは [CreateKey](#) オペレーションの [タグ](#) パラメータを使用して、KMS キー作成時に KMS キーにタグを追加できます。

JSON

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "IAMPolicyCreateKeys",
    "Effect": "Allow",
    "Action": "kms:CreateKey",
    "Resource": "*"
  },
  {
    "Sid": "IAMPolicyTags",
    "Effect": "Allow",
    "Action": [
      "kms:TagResource",
      "kms:UntagResource",
      "kms:ListResourceTags"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/*"
  }
]
```

タグ付け許可を制限する

[ポリシー条件](#)を使用して、タグ付け許可を制限できます。次のポリシー条件を `kms:TagResource` および `kms:UntagResource` 許可に適用できます。例えば、`aws:RequestTag/tag-key` 条件を使用して、プリンシパルが特定のタグのみを追加できるようにするか、プリンシパルが特定のタグキーを持つタグを追加しないように許可できます。または、`kms:KeyOrigin` 条件を使用して、プリンシパルが [インポートされたキーマテリアルを持つ](#) KMS キーにタグ付けまたはタグ解除を行わないようにすることができます。

- [aws:RequestTag](#)
- [aws:ResourceTag/tag-key](#) (IAM ポリシーのみ)
- [aws:TagKeys](#)
- [kms:CallerAccount](#)
- [kms:KeySpec](#)
- [kms:KeyUsage](#)
- [kms:KeyOrigin](#)
- [kms:ViaService](#)

ベストプラクティスとして、タグを使用して KMS キーへのアクセスを制御する場合は、`aws:RequestTag/tag-key` または `aws:TagKeys` 条件キーを使用して、許可するタグ (またはタグキー) を決定します。

例えば、次の IAM ポリシーは前述のものと似ています。ただしこのポリシーでは、プリンシパルはタグ (TagResource) の作成とタグ UntagResource の削除を、Project タグキーを持つタグに対してのみ実行できます。

TagResource および UntagResource リクエストは複数のタグを含む可能性があるため、ForAllValues または ForAnyValue 集合演算子を[aws:TagKeys](#) 条件で指定する必要があります。ForAnyValue 演算子では、リクエスト内のタグキー 1 つ以上が、ポリシーのタグキーの 1 つと一致する必要があります。ForAllValues 演算子では、リクエスト内のタグキーすべてが、ポリシーのタグキーの 1 つと一致する必要があります。ForAllValues 演算子はリクエストにタグがない場合も true を返しますが、TagResource と UntagResource は、タグが指定されていない場合、失敗します。集合演算子の詳細については、「IAM ユーザーガイド」の「[複数のキーと値の使用](#)」を参照してください。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",
      "Effect": "Allow",
      "Action": "kms:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyViewAllTags",
      "Effect": "Allow",
      "Action": "kms:ListResourceTags",
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPolicyManageTags",
      "Effect": "Allow",
      "Action": [
        "kms:TagResource",
        "kms:UntagResource"
      ]
    }
  ]
}
```

```
"Resource": "arn:aws:kms:*:111122223333:key/*",
"Condition": {
  "ForAllValues:StringEquals": {"aws:TagKeys": "Project"}
}
}
```

KMS キーにタグを追加する

タグは、AWS リソースを識別して整理するのに役立ちます。[KMS キーを作成](#)する際にカスターマネージドキーにタグを追加したり、あるいは既存の KMS キーにタグを追加することができます。タグを付けることはできません AWS マネージドキー。

次の手順は、コンソールと AWS KMS API を使用して AWS KMS カスターマネージドキーにタグを追加する方法を示しています。AWS KMS API の例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用できます。

トピック

- [KMS キーの作成中にタグを追加する](#)
- [既存の KMS キーにタグを追加する](#)

KMS キーの作成中にタグを追加する

AWS KMS コンソールまたは [CreateKey](#) オペレーションを使用してキーを作成するときに、KMS キーにタグを追加できます。KMS キーの作成中にタグを追加するには、KMS キーの作成に必要なアクセス許可に加えて、IAM ポリシー内での kms:TagResource アクセス許可が必要になります。少なくとも、アクセス許可はアカウントとリージョン内のすべての KMS キーを対象にする必要があります。詳細については、「[タグへのアクセスを制御する](#)」を参照してください。

AWS KMS コンソールの使用

コンソールでの KMS キー作成中にタグを追加するには、KMS キーにタグ付けして作成するのに必要なアクセス許可に加えて、コンソール内で KMS キーを表示するのに必要なアクセス許可も必要になります。少なくとも、アクセス許可はアカウントとリージョン内のすべての KMS キーを対象にする必要があります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。(AWS マネージドキーのタグを管理することはできません)
4. キータイプを選択し、[Next] (次へ) を選択します。
5. エイリアスおよび説明 (オプション) を入力します。
6. タグキーおよびタグ値 (オプション) を入力します。タグを追加するには、[Add tag] (タグの追加) を選択します。タグを削除するには、[Remove] (削除) を選択します。新しい KMS キーのタグ付けが完了したら、[Next] (次へ) を選択します。
7. KMS キーの作成を完了します。

AWS KMS API の使用

[CreateKey](#) オペレーションを使用してキーを作成する際にタグを指定するには、オペレーションの Tags パラメータを使用します。

Tags パラメータ値の CreateKey は、大文字と小文字を区別するタグキーとタグ値のペアのコレクションです。KMS キーのそれぞれのタグは、異なるタグ名を持つ必要があります。タグ値は、NULL または空の文字列にすることができます。

たとえば、次の AWS CLI コマンドは、Project:Alpha タグを使用して対称暗号化 KMS キーを作成します。複数のキーと値のペアを指定する場合は、スペースを使用して各ペアを区切ります。

```
$ aws kms create-key --tags TagKey=Project,TagValue=Alpha
```

このコマンドが成功すると、新しい KMS キーに関する情報を含む KeyMetadata オブジェクトが返されます。ただし、KeyMetadata にはタグは含まれません。タグを取得するには、[ListResourceTags](#) オペレーションを使用します。

既存の KMS キーにタグを追加する

AWS KMS コンソールで、または [TagResource](#) オペレーションを使用して、既存のカスタマーマネージド KMS キーにタグを追加できます。タグを追加するには、KMS キーに対するタグ付けアクセス許可が必要です。この許可は、KMS キーのキーポリシーから取得できます。または、キーポリシーで許可されている場合、KMS キーを含む IAM ポリシーから取得できます。

AWS KMS コンソールの使用

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタマーマネージドキー] を選択します。(AWS マネージドキーのタグを管理することはできません)
4. テーブルフィルターを使用して、特定のタグを持つ KMS キーのみを表示できます。詳細については、「[AWS KMS コンソールを使用してタグを表示する](#)」を参照してください。
5. KMS キーのエイリアスの横にあるチェックボックスをオンにします。
6. [Key actions]、[Add or edit tags] の順に選択します。
7. KMS キーの詳細ページで、[Tags] (タグ) タブを選択します。
 - 最初のタグを作成するには、[Create tag] (タグの作成) を選択し、タグキー (必須) とタグ値 (オプション) を入力して、[Save] (保存) を選択します。

タグ値を空白のままにすると、実際のタグ値は NULL または空の文字列になります。

 - タグを追加するには、[Edit] (編集)、[Add tag] (タグの追加) の順に選択し、タグキーとタグ値を入力して、[Save] (保存) を選択します。
8. 変更を保存するには、変更の保存を選択します。

AWS KMS API の使用

[TagResource](#) オペレーションは KMS キーに 1 つ以上のタグを追加します。このオペレーションを使用して、別の AWS アカウント内でタグを追加することはできません。TagResource オペレーションを使用して既存のタグを編集することもできます。詳細については、「[the section called “タグの編集”](#)」を参照してください。

タグを追加するには、新しいタグキーとタグ値を指定します。KMS キーのそれぞれのタグは、異なるタグキーを持つ必要があります。タグ値は、NULL または空の文字列にすることができます。

例えば、次のコマンドでは、サンプルの KMS キーに **Purpose** タグおよび **Department** タグを追加します。

```
$ aws kms tag-resource \  
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
    --tags TagKey=Purpose,TagValue=Pretest TagKey=Department,TagValue=Finance
```

このコマンドが成功した場合、出力を返しません。タグを KMS キーで表示するには、[ListResourceTags](#) オペレーションを使用します。

KMS キーに関連付けられたタグを編集する

タグは、AWS リソースを識別して整理するのに役立ちます。カスターマネージド KMS キーに関連付けられたタグは、AWS KMS コンソールで編集するか、[TagResource](#) オペレーションを使用して編集できます。のタグを編集することはできません AWS マネージドキー。

以下の手順は、KMS キーに関連付けられたタグを編集する方法を示します。この AWS KMS API 例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用できます。

AWS KMS コンソールの使用

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。(のタグは編集できません AWS マネージドキー)
4. テーブルフィルタを使用して、特定のタグを持つ KMS キーのみを表示できます。詳細については、「[AWS KMS コンソールを使用してタグを表示する](#)」を参照してください。
5. KMS キーのエイリアスの横にあるチェックボックスをオンにします。
6. [Key actions]、[Add or edit tags] の順に選択します。
7. KMS キーの詳細ページで、[Tags] (タグ) タブを選択します。
 - タグの名前または値を変更するには、[Edit (編集)] を選択して変更を加えた後、[Save (保存)] を選択します。
8. 変更を保存するには、変更の保存を選択します。

AWS KMS API の使用

[TagResource](#) オペレーションは、カスターマネージドキーに 1 つ以上のタグを追加します。ただし、TagResource を使用して既存のタグのタグ値を変更することもできます。このオペレーションを使用して、別の AWS アカウントのタグを追加または編集することはできません。

タグを編集するには、既存のタグキーと新しいタグ値を指定します。KMS キーのそれぞれのタグは、異なるタグキーを持つ必要があります。タグ値は、NULL または空の文字列にすることができます。

例えば、このコマンドは Purpose タグの値 Pretest をからに変更します Test。

```
$ aws kms tag-resource \  
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
    --tags TagKey=Purpose,TagValue=Test
```

KMS キーに関連付けられたタグを削除する

タグは、AWS リソースを識別して整理するのに役立ちます。カスターマネージド KMS キーに関連付けられたタグは、AWS KMS コンソールで、または [UntagResource](#) オペレーションを使用して削除できます。のタグを編集または削除することはできません AWS マネージドキー。

以下の手順は、KMS キーからタグを削除する方法を示します。この AWS KMS API 例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用できます。

AWS KMS コンソールの使用

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。(AWS マネージドキーのタグを管理することはできません)
4. テーブルフィルターを使用して、特定のタグを持つ KMS キーのみを表示できます。詳細については、「[AWS KMS コンソールを使用してタグを表示する](#)」を参照してください。
5. KMS キーのエイリアスの横にあるチェックボックスをオンにします。
6. [Key actions]、[Add or edit tags] の順に選択します。
7. KMS キーの詳細ページで、[Tags] (タグ) タブを選択します。
 - タグを削除するには、[Edit (編集)] を選択します。そのタグの行で、[Remove (削除)] を選択してから [Save (保存)] を選択します。
8. 変更を保存するには、変更の保存を選択します。

AWS KMS API の使用

[UntagResource](#) オペレーションでは、KMS キーからタグを削除します。削除するタグを識別するには、タグキーを指定します。このオペレーションを使用して、別の AWS アカウントの KMS キーからタグを削除することはできません。

成功すると、UntagResource オペレーションは出力を返しません。また、指定したタグキーが KMS キーで見つからない場合、例外をスローしたり、レスポンスを返したりすることはありません。オペレーションが正常に機能したことを確認するには、[ListResourceTags](#) オペレーションを使用します。

例えば、このコマンドでは、指定した KMS キーから **Purpose** タグとその値を削除します。

```
$ aws kms untag-resource --key-id 1234abcd-12ab-34cd-56ef-1234567890ab --tag-keys Purpose
```

KMS キーに関連付けられたタグを表示する

タグは、AWS リソースを識別して整理するのに役立ちます。カスターマネージド KMS キーに関連付けられたタグは、AWS KMS コンソールで、または [ListResourceTags](#) オペレーションを使用して表示できます。

次の手順は、特定の KMS キーに関連付けられたタグを検索する方法を示します。AWS KMS API の例では [AWS Command Line Interface \(AWS CLI\)](#) を使用していますが、サポートされている任意のプログラミング言語を使用できます。

AWS KMS コンソールの使用

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスターマネージドキー] を選択します。(AWS マネージドキーのタグを管理することはできません)
4. テーブルフィルターを使用して、特定のタグを持つ KMS キーのみを表示できます。

特定のタグを持つ KMS キーのみを表示するには、フィルターボックス、タグキーの順に選択し、実際のタグ値の中から選択します。タグ値のすべてまたは一部を入力することもできます。

結果のテーブルには、選択したタグを持つすべての KMS キーが表示されます。ただし、タグは表示されません。タグを表示するには、KMS キーのキー ID またはエイリアスを選択し、その詳細ページで [Tags] (タグ) タブを選択します。これらのタブは、一般設定セクションにあります。

このフィルターには、タグキーとタグ値の両方が必要です。タグキーのみを入力しても、その値のみを入力しても KMS キーは検索されません。タグキー、値のすべてまたは一部でタグをフィルタリングするには、[ListResourceTags](#) オペレーションを使用してタグ付けされた KMS キーを取得し、プログラミング言語のフィルタリング機能を使用します。

5. KMS キーのエイリアスの横にあるチェックボックスをオンにします。
6. [Key actions]、[Add or edit tags] の順に選択します。
7. KMS キーの詳細ページで、[Tags] (タグ) タブを選択します。

AWS KMS API の使用

[ListResourceTags](#) オペレーションでは、KMS キーのタグを取得します。KeyId パラメータは必須です。このオペレーションを使用して、別の AWS アカウントの KMS キーのタグを表示することはできません。

例えば、次のコマンドでは、サンプルの KMS キーのタグを取得します。

```
$ aws kms list-resource-tags --key-id 1234abcd-12ab-34cd-56ef-1234567890ab

{"Truncated": false,
 "Tags": [
   {
     "TagKey": "Project",
     "TagValue": "Alpha"
   },
   {
     "TagKey": "Purpose",
     "TagValue": "Test"
   },
   {
     "TagKey": "Department",
     "TagValue": "Finance"
   }
 ]
}
```

タグを使用して KMS キーへのアクセスを制御する

KMS キーのタグ AWS KMS keys に基づいて へのアクセスを制御できます。例えば、プリンシパルが特定のタグを持つ KMS キーのみを有効または無効にすることを許可する IAM ポリシーを書き込むことができます。または、IAM ポリシーを使用して、KMS キーに特定のタグがない限り、プリンシパルが暗号化オペレーションで KMS キーを使用しないようにすることもできます。

この機能は、[属性ベースのアクセスコントロール \(ABAC\)](#) AWS KMS のサポートの一部です。タグを使用してリソースへのアクセス AWS を制御する方法については、IAM ユーザーガイドの「[ABAC for AWSとは](#)」および「[リソースタグを使用した AWS リソースへのアクセスの制御](#)」を参照してください。ABAC に関連するアクセス問題の解決方法については、[の ABAC のトラブルシューティング AWS KMS](#) を参照してください。

Note

タグとエイリアスの変更が KMS キーの認可に影響を及ぼすまでに最長 5 分かかることがあります。最近の変更は、認可に影響を与える前に API オペレーションで表示される場合があります。

AWS KMS は [aws:ResourceTag/tag-key グローバル条件コンテキストキー](#) をサポートしています。これにより、KMS キーのタグに基づいて KMS キーへのアクセスを制御できます。複数の KMS キーに同じタグを付けることができるため、この機能を使用すると、KMS キーの選択したセットにアクセス許可を適用できます。それらのタグを変更することで、セット内の KMS キーを簡単に変更することもできます。

では AWS KMS、[aws:ResourceTag/tag-key](#) 条件キーは IAM ポリシーでのみサポートされています。これは、1 つの KMS キーだけに適用されるキーポリシー、または特定の KMS キーを使用しないオペレーション ([ListKeys](#) または [ListAliases](#) オペレーションなど) ではサポートされません。

タグを使用してアクセスを制御すると、シンプルでスケーラブルかつ柔軟な方法でアクセス許可を管理できます。ただし、適切に設計および管理されていない場合、KMS キーへのアクセスを誤って許可または拒否する可能性があります。タグを使用してアクセスを制御する場合は、次の方法を検討します。

- タグを使用して、[最小特権アクセス](#)のベストプラクティスを強化します。IAM プリンシパルで、使用または管理する必要がある KMS キーのみに対して、必要なアクセス許可のみを付与します。例えば、タグを使用して、プロジェクトに使用する KMS キーにラベルを付けます。次に、プロジェクトタグで KMS キーのみを使用する許可をプロジェクトチームに付与します。

- プリンシパルにタグを追加、編集、削除できる `kms:TagResource` および `kms:UntagResource` 許可を付与する際は注意してください。タグを使用して KMS キーへのアクセスを制御する場合、タグを変更すると、使用許可のない KMS キーに対する使用許可をプリンシパルに付与してしまう可能性があります。他のプリンシパルがジョブを実行するために必要な KMS キーへのアクセスを拒否することもできます。キーポリシーを変更したり、権限を作成したりする許可を持たないキー管理者も、タグを管理する許可があれば、KMS キーへのアクセスを制御できます。

可能な限り、ポリシー条件 (`aws:RequestTag/tag-key` または `aws:TagKeys` ~ [プリンシパルのタグ付け許可の制限](#) など) を、特定の KMS キーの特定のタグまたはタグパターンに適用します。
- 現在タグ付けおよびタグ付け解除 AWS アカウント のアクセス許可を持っている のプリンシパルを確認し、必要に応じて調整します。例えば、コンソールの [キー管理者のデフォルトキーポリシー](#) は、KMS キーの `kms:TagResource` および `kms:UntagResource` アクセス許可を含んでいます。IAM ポリシーでは、すべての KMS キーに対してタグ付けおよびタグ解除を許可する場合があります。例えば、[AWSKeyManagementServicePowerUser](#) マネージドポリシーは、プリンシパルがすべての KMS キーで、タグ付け、タグ解除、タグの一覧表示を行うことを許可します。
- タグに依存するポリシーを設定する前に、 の KMS キーのタグを確認してください AWS アカウント。含めるタグにのみポリシーが適用されることを確認します。[CloudTrail ログ](#) および [CloudWatch アラーム](#) を使用して、KMS キーへのアクセスに影響する可能性のある変更をタグ付けするようにアラートします。
- タグベースのポリシー条件では、パターンマッチングを使用します。タグの特定のインスタンスには関連付けられません。タグベースの条件キーを使用するポリシーは、パターンに一致するすべての新規および既存のタグに影響します。ポリシー条件に一致するタグを削除して再作成すると、古いタグの場合と同様に、新しいタグに条件が適用されます。

例えば、次の IAM ポリシーの例を考えてみます。アカウントがアジアパシフィック (シンガポール) リージョンで "Project"="Alpha" タグを持つ KMS キーのみで、[GenerateDataKeyWithoutPlaintext](#) および [Decrypt](#) オペレーションを呼び出すことを、プリンシパルに許可します。このポリシーは、サンプルの Alpha プロジェクトでロールにアタッチできます。

JSON

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

```

"Statement": [
  {
    "Sid": "IAMPolicyWithResourceTag",
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKeyWithoutPlaintext",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:ap-southeast-1:111122223333:key/*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/Project": "Alpha"
      }
    }
  }
]
}

```

次の IAM ポリシーの例では、プリンシパルが、特定の暗号化オペレーションのために、アカウントで任意の KMS キーを使用することを許可します。ただし、プリンシパルが "Type"="Reserved" タグのある、または "Type" タグのない KMS キーにこれらの暗号化オペレーションを使用することを禁止します。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAllowCryptographicOperations",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey*",
        "kms:Decrypt",
        "kms:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*"
    },
    {
      "Sid": "IAMDenyOnTag",

```



```
"Effect": "Deny",
"Action": [
  "kms:Encrypt",
  "kms:GenerateDataKey*",
  "kms:Decrypt",
  "kms:ReEncrypt*"
],
"Resource": "arn:aws:kms:*:111122223333:key/*",
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/Type": "Reserved"
  }
},
{
  "Sid": "IAMDenyNoTag",
  "Effect": "Deny",
  "Action": [
    "kms:Encrypt",
    "kms:GenerateDataKey*",
    "kms:Decrypt",
    "kms:ReEncrypt*"
  ],
  "Resource": "arn:aws:kms:*:111122223333:key/*",
  "Condition": {
    "Null": {
      "aws:ResourceTag/Type": "true"
    }
  }
}
]
```

キーストア

キーストアは、暗号化キーを保存し使用するための安全な場所です。のデフォルトキーストアは、保存するキーを生成および管理する方法 AWS KMS もサポートしています。デフォルトでは、で AWS KMS keys 作成する の暗号化キーマテリアル AWS KMS は で生成され、[FIPS 140-3 暗号化モジュール検証プログラム](#)であるハードウェアセキュリティモジュール (HSMs) によって保護されます。KMS キーのキーマテリアルは、暗号化されずに HSM の外に出ることはありません。

AWS KMS は、を使用して AWS KMS 暗号化キーを作成および管理するときにキーマテリアルを保護するために、いくつかのタイプのキーストアをサポートしています。が提供するすべてのキーストアオプション AWS KMS は、セキュリティレベル 3 の FIPS 140-3 で継続的に検証され、AWS オペレーターを含むすべてのユーザーがアクセス許可なしでプレーンテキストキーにアクセスしたり、使用したりできないように設計されています。

AWS KMS 標準キーストア

デフォルトでは、KMS キーは標準の AWS KMS HSM を使用して作成されます。この HSM タイプは、お客様にとって最も拡張性があり、最も低コストで管理しやすいキーストアを提供する、HSM のマルチテナントフリートと考えることができます。サービスがユーザーに代わってデータを暗号化 AWS のサービス できるように、1 つ以上の 内で使用するために KMS キーを作成する場合は、対称キーを作成します。独自のアプリケーション設計向けに KMS キーを使用する場合は、対称暗号化キー、非対称キー、または HMAC キーのいずれかを選択して作成できます。

標準キーストアオプションでは、はキー AWS KMS を作成し、サービスが内部で管理するキーで暗号化します。その後、暗号化されたバージョンのキーの複数のコピーが、耐久性を考慮して設計されたシステムに保存されます。標準キーストアタイプでキーマテリアルを生成して保護することで、キーストアの運用上の負担とコストを最小限に抑え AWS KMS ながら、の AWS スケーラビリティ、可用性、耐久性を最大限に活用できます。

AWS KMS インポートされたキーマテリアルを持つ標準キーストア

特定のキーの唯一のコピーの生成と保存の両方 AWS KMS を要求する代わりに、にキーマテリアルをインポートすることを選択できます。これにより AWS KMS、独自の 256 ビット対称暗号化キー、RSA または楕円曲線 (ECC) キー、またはハッシュベースのメッセージ認証コード (HMAC) キーを生成し、KMS キー識別子 (keyId) に適用できます。これは、Bring Your Own Key (BYOK) 方式と呼ばれることがあります。ローカルキー管理システムからインポートされたキーマテリアルは、

によって発行されたパブリックキー AWS KMS、サポートされている暗号化ラップアルゴリズム、および によって提供される時間ベースのインポートトークンを使用して保護する必要があります AWS KMS。このプロセスにより、暗号化されてインポートされたキーは、所有ユーザーの環境を離れた後のみ AWS KMS HSM による復号化が可能になります。

インポートされたキーマテリアルは、キーを生成するシステムに関する特定の要件がある場合、または の外部でキーのコピーをバックアップ AWS として必要とする場合に便利です。インポートされたキーマテリアルの全般的な可用性と耐久性については、ユーザー自身の責任となることにご留意ください。AWS KMS にはインポートされたキーのコピーがあり、必要とされる間は高可用性を維持しますが、インポートされたキーは、削除のための特別な API「DeleteImportedKeyMaterial」を提供します。この API は、AWS KMS がキーを復元するオプションなしで、インポートされたキーマテリアルのすべてのコピー AWS をすぐに削除します。また、インポートされたキーの有効期限を設定することができ、有効期限到達後はそのキーは使用できなくなります。キーを再度有効にするには AWS KMS、キーマテリアルを再インポートし、同じ keyId に割り当てる必要があります。インポートされたキーに対するこの削除アクションは、 がユーザーに代わって AWS KMS 生成して保存する標準キーとは異なります。標準の場合、キー削除プロセスには強制的待機期間があり、削除が予定されているキーはまず使用できないようにブロックされます。このアクションにより、データにアクセスするためにそのキーを必要とする可能性のあるアプリケーションまたは AWS サービスのログでアクセス拒否エラーを確認できます。このようなアクセスリクエストが表示された場合は、スケジュールされた削除をキャンセルしてキーを再度有効にすることができます。設定可能な待機期間 (7~30 日) が経過すると、KMS はキーマテリアル、keyID、およびキーに関連付けられたすべてのメタデータを実際に削除します。可用性と耐久性の詳細については、「[AWS KMS デベロッパーガイド](#)」の「[インポートされたキーマテリアルの保護](#)」を参照してください。

インポートされたキーマテリアルについては、留意すべき追加の制限事項があります。AWS KMS は新しいキーマテリアルを生成できないため、インポートされたキーの自動ローテーションを設定する方法はありません。有効なローテーションを実現するためには、新しい keyId を使用して新しい KMS キーを作成し、そのうえで新しいキーマテリアルをインポートする必要があります。また、インポートされた対称キー AWS KMS で作成された暗号文は、 の外部にあるキーのローカルコピーを使用して簡単に復号化することはできません AWS。これは、 が使用する認証された暗号化形式 AWS KMS が暗号文にメタデータを追加して、復号オペレーション中に、以前の暗号化オペレーションで想定される KMS キーによって暗号文が作成されたことを保証するためです。ほとんどの外部暗号化システムは、こうしたメタデータを解析して未加工の暗号文にアクセスし、対称キーのコピーを使用可能にする方法を理解していません。によって暗号化テキストに追加されるメタデータ AWS KMS がないため、インポートされた非対称キー (RSA や ECC など) で作成された暗号化テキストは、キーの一致する部分 (パブリックまたはプライベート) AWS KMS で の外部で使用できます。

AWS KMS カスタムキーストア

ただし、HSM の管理をさらに強化する必要がある場合は、カスタムキーストアを作成します。

カスタムキーストアは、内のキーストア AWS KMS で AWS KMS、ユーザーが所有および管理している 外のキーマネージャーによってバックアップされます。カスタムキーストアは、の便利で包括的なキー管理インターフェイス AWS KMS と、キーマテリアルと暗号化オペレーションを所有および制御する機能を組み合わせます。カスタムキーストアで KMS キーを使用する場合、暗号化のオペレーションは、ユーザーのキーマネージャーが、ユーザーの暗号化キーを使用して実行します。それにより、暗号化キーの可用性と耐久性、および HSM のオペレーションに対するユーザーの責任が増えます。

ユーザーによる HSM の所有は、標準 KMS キーストアなどのマルチテナントウェブサービスによる暗号化キーの保持を認めていない一部の法規制要件を満たすのに役立ちます。カスタムキーストアは AWS、マネージド HSMs を使用する KMS キーストアよりも安全ではありませんが、管理とコストへの影響は異なります (それ以上)。その結果、暗号化キーの可用性と耐久性、および HSM のオペレーションに対するユーザーの責任が増えます。AWS KMS HSMs で標準キーストアを使用するか、カスタムキーストアを使用するかにかかわらず、このサービスは、AWS 従業員を含め、誰もアクセス許可なしでプレーンテキストキーを取得したり使用したりできないように設計されています。は、2 種類のカスタムキーストア、AWS CloudHSM キーストア、外部キーストア AWS KMS をサポートしています。

サポートされていない機能

AWS KMS は、カスタムキーストアで以下の機能をサポートしていません。

- [非対称 KMS キー](#)
- [HMAC KMS キー](#)
- [インポートされたキーマテリアルを持つ KMS キー](#)
- [自動キーローテーション](#)
- [マルチリージョンキー](#)

AWS CloudHSM キーストア

[AWS CloudHSM](#) キーストアに KMS キーを作成できます。ここでは、ルートユーザーキーが生成、保存され、所有および管理している AWS CloudHSM クラスターで使用されます。一部の暗号化オペレーションにキーを使用する AWS KMS へのリクエストは、オペレーションを実行するために AWS CloudHSM クラスターに転送されます。AWS CloudHSM クラスターが によってホストされている

間は AWS、ユーザーが直接管理および運用するシングルテナントソリューションです。お客様は、AWS CloudHSM クラスター内の KMS キーの可用性とパフォーマンスの大部分を所有しています。AWS CloudHSM カスタムキーストアが要件に適しているかどうかを確認するには、AWS セキュリティブログの「[カスタムキーストアは適切ですか？](#)」を参照してください [AWS KMS](#)。

外部キーストア

外部キーストア (XKS) を使用する AWS KMS ように を設定できます。ここで、ルートユーザーキーは、外のキー管理システムで生成、保存、使用されます AWS クラウド。一部の暗号化オペレーションでキーを使用するための AWS KMS へのリクエストは、ユーザーの外部ホストシステムに転送されてそのオペレーションの実行に使用されます。具体的には、リクエストはまずネットワーク内の XKS プロキシに転送され、さらに XKS プロキシから使用する暗号化システムへと転送されます。XKS プロキシは、誰でも統合できるオープンソース仕様です。多くの商用キー管理ベンダーは XKS プロキシ仕様をサポートしています。外部キーストアはお客様または第三者によってホストされるため、システム内のキーの可用性、耐久性、およびパフォーマンスはすべてユーザーの責任となります。外部キーストアが要件に適しているかどうかを確認するには、AWS ニュースブログの[AWS KMS 「外部キーストア \(XKS\) の発表」](#)を参照してください。

AWS CloudHSM キーストア

AWS CloudHSM キーストアは、[AWS CloudHSM クラスター](#)によってバックアップされた[カスタムキーストア](#)です。カスタムキーストア AWS KMS key で を作成すると、 は、ユーザーが所有および管理する AWS CloudHSM クラスターに KMS キーの抽出不可能なキーマテリアル AWS KMS を生成して保存します。カスタムキーストアで KMS キーを使用する際、[暗号化オペレーション](#)はクラスター内の HSM で実行されます。この機能は、 の利便性と広範な統合 AWS KMS と、 内の AWS CloudHSM クラスターの追加の制御を組み合わせたものです AWS アカウント。

AWS KMS は、カスタムキーストアの作成、使用、管理のためのコンソールと API の完全なサポートを提供します。任意の KMS キーを使用するのと同じ方法で、カスタムキーストア内の KMS キーを使用できます。例えば、KMS キーを使用して、データキーの生成やデータの暗号化を実行できます。カスタマーマネージドキーをサポートする AWS サービスで、カスタムキーストアの KMS キーを使用することもできます。

カスタムキーストアは必要ですか？

ほとんどのユーザーにとって、[FIPS 140-3 検証済み暗号化モジュール](#)で保護されているデフォルトの AWS KMS キーストアは、セキュリティ要件を満たします。メンテナンス責任や追加サービスへの依存の強化は必要ありません。

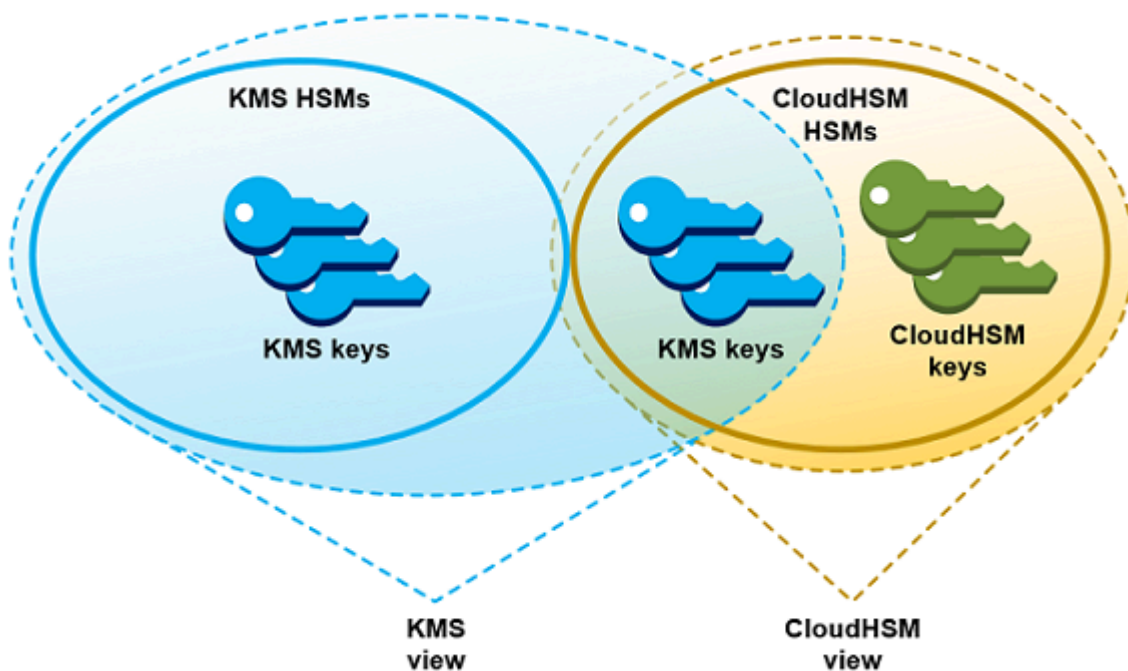
ただし、組織に次のいずれかの要件がある場合、カスタムキーストアの作成を検討してください。

- 単一テナント HSM または直接制御できる HSM における保護が明示的に要求されているキーがある。
- キーマテリアルをすぐに削除できる必要があります AWS KMS。
- AWS KMS または とは別に、キーの使用をすべて監査できる必要があります AWS CloudTrail。

カスタムキーストアはどのように機能しますか？

各カスタムキーストアは、の AWS CloudHSM クラスターに関連付けられています AWS アカウント。カスタムキーストアをクラスターに接続すると、は接続をサポートするネットワークインフラストラクチャ AWS KMS を作成します。次に、クラスター内の[専用 Crypto User](#) の認証情報を使用して、クラスター内のキー AWS CloudHSM クライアントにログインします。

でカスタムキーストアを作成および管理 AWS KMS し、で HSM クラスターを作成および管理します AWS CloudHSM。AWS KMS カスタムキーストア AWS KMS keys でを作成するときは、で KMS キーを表示および管理します AWS KMS。ただし、クラスターの他のキーに実行するのと同じように、AWS CloudHSMでキーマテリアルを表示して管理することもできます。



カスタムキーストアによって生成されたキーマテリアルを使用して、[対称暗号化 KMS](#) キーを作成できます。AWS KMS 次に、同じ方法を使用して、キーストアの KMS キーに使用するカスタムキー AWS KMS ストアの KMS キーを表示および管理します。IAM ポリシーとキーポリシーを使用してア

アクセスを制御したり、タグやエイリアスを作成したり、KMS キーを有効および無効にしたり、キーの削除をスケジュールリングしたりできます。[暗号化オペレーション](#)に KMS キーを使用し、と統合する AWS サービスで使用できます AWS KMS。

さらに、HSMs の作成と削除、バックアップの管理など、AWS CloudHSM クラスターを完全に制御できます。AWS CloudHSM クライアントおよびサポートされているソフトウェアライブラリを使用して、KMS キーのキーマテリアルを表示、監査、管理できます。カスタムキーストアが切断されている間は、AWS KMS はカスタムキーストアにアクセスできず、ユーザーはカスタムキーストアの KMS キーを暗号化オペレーションに使用することはできません。こうした制御の強化により、カスタムキーストアがそれを必要とする組織のための強力なソリューションとなります。

開始方法

AWS CloudHSM キーストアを作成および管理するには、AWS KMS および の機能を使用します AWS CloudHSM。

1. 開始します AWS CloudHSM。[アクティブな AWS CloudHSM クラスターを作成するか](#)、既存のクラスターを選択します。クラスターには、少なくとも 2 つの異なるアベイラビリティゾーンのアクティブな HSM が必要です。次に、AWS KMS のクラスターで[専用 crypto user \(CU\) アカウント](#)を作成します。
2. で AWS KMS、選択した AWS CloudHSM クラスターに関連付けられている[カスタムキーストアを作成します](#)。は、カスタムキーストアを作成、表示、編集、削除できる完全な管理インターフェイス AWS KMS を提供します。
3. カスタムキーストアを使用する準備ができれば、[関連付けられた AWS CloudHSM クラスターに接続します](#)。は、接続をサポートするために必要なネットワークインフラストラクチャ AWS KMS を作成します。次に、専用 crypto user アカウント認証情報を使用してクラスターへのログインが行われ、クラスターでキーマテリアルを生成して管理できるようになります。
4. これで、[カスタムキーストアで対称暗号化 KMS キーを作成](#)できるようになりました。カスタムキーストアを指定するだけで、KMS キーを作成できます。

どの時点で問題が発生しても、「[カスタムキーストアのトラブルシューティング](#)」トピックでヘルプを見つけることができます。回答が見つからない場合は、このガイドの各ページの下部にあるフィードバックリンクを使用するか、または [AWS Key Management Service ディスカッションフォーラム](#)に質問を投稿してください。

クォータ

AWS KMS では、接続状態に関係なく、[キーストアと外部キーストアの両方を含め、各 リージョンに最大 10 個のカスタムキーストア](#)を使用できます。AWS アカウント [AWS CloudHSM](#) さらに、[キーストアでの KMS AWS CloudHSM キーの使用](#)にはリクエストクォータがあります AWS KMS。

料金表

AWS KMS カスタムキーストアとカスタムキーストアのカスタマーマネージドキーのコストについては、「[の AWS Key Management Service 料金](#)」を参照してください。AWS CloudHSM クラスターと HSMs」を参照してください。 [AWS CloudHSM](#)

リージョン

AWS KMS AWS KMS は、アジアパシフィック (メルボルン)、中国 (北京)、中国 (寧夏)、欧州 (スペイン) を除く、AWS リージョン がサポートされているすべての AWS CloudHSM キーストアをサポートします。

サポートされていない機能

AWS KMS は、カスタムキーストアで以下の機能をサポートしていません。

- [非対称 KMS キー](#)
- [HMAC KMS キー](#)
- [インポートされたキーマテリアルを持つ KMS キー](#)
- [自動キーローテーション](#)
- [マルチリージョンキー](#)

AWS CloudHSM キーストアの概念

このトピックでは、AWS CloudHSM キーストアで使用される用語と概念について説明します。

AWS CloudHSM キーストア

AWS CloudHSM キーストアは、所有および管理する クラスターに関連付けられた[カスタムキーストア](#)です。AWS CloudHSM クラスターは、AWS CloudHSM [FIPS 140-2 または FIPS 140-3 Level 3](#) で認定されたハードウェアセキュリティモジュール (HSMs) によってバックアップされます。

AWS CloudHSM キーストアで KMS キーを作成すると、は関連付けられた AWS CloudHSM クラスターに 256 ビットの永続的でエクスポート不可能な Advanced Encryption Standard (AES) 対称キー

AWS KMS を生成します。このキーマテリアルは、HSM を非暗号化のままにしません。キーストアで KMS AWS CloudHSM キーを使用すると、暗号化オペレーションはクラスターの HSMs で実行されます。

AWS CloudHSM キーストアは、の便利で包括的なキー管理インターフェイス AWS KMS と、の AWS CloudHSM クラスターによって提供される追加のコントロールを組み合わせます AWS アカウント。この統合機能により、クラスター、HSM、バックアップの管理など、キーマテリアルを保存する HSMs を完全に制御 AWS KMS しながら、で HSMs キーを作成、管理、使用できます。AWS KMS コンソールと APIs を使用して、キーストアとその KMS キーを管理 AWS CloudHSM できます。AWS CloudHSM コンソール、APIs、クライアントソフトウェア、および関連するソフトウェアライブラリを使用して、関連するクラスターを管理することもできます。

AWS CloudHSM キーストア [の表示と管理](#)、[プロパティの編集](#)、関連する AWS CloudHSM クラスター [への接続と切断](#)を行うことができます。[キーストアを削除する必要がある場合は AWS CloudHSM](#)、まず AWS CloudHSM キーストア内の KMS キーを削除する必要があります。そのために、削除をスケジュールし、猶予期間が終了するまで待機します。キーストアを削除する AWS CloudHSM と、からリソースが削除されますが AWS KMS、クラスターには影響 AWS CloudHSM しません。

AWS CloudHSM クラスター

すべての AWS CloudHSM キーストアは 1 つの AWS CloudHSM クラスターに関連付けられています。AWS CloudHSM キーストア AWS KMS key でを作成すると、AWS KMS は関連付けられたクラスターにキーマテリアルを作成します。AWS CloudHSM キーストアで KMS キーを使用すると、関連付けられたクラスターで、暗号化オペレーションが実行されます。

各 AWS CloudHSM クラスターは、1 つの AWS CloudHSM キーストアにのみ関連付けることができます。選択したクラスターを別の AWS CloudHSM キーストアに関連付けることはできません。また、バックアップ履歴を別の AWS CloudHSM キーストアに関連付けられているクラスターと共有することはできません。クラスターは初期化されてアクティブである必要があります、キーストアと同じ AWS アカウント およびリージョン AWS CloudHSM にある必要があります。新しいクラスターを作成することも、既存のクラスターを使用することもできます。クラスターを排他的に使用する AWS KMS 必要はありません。AWS CloudHSM キーストアで KMS キーを作成するには、関連付けられたクラスターに少なくとも 2 つのアクティブな HSMs が含まれている必要があります。他のすべてのオペレーションでは、1 つの HSM しか必要ありません。

AWS CloudHSM キーストアの作成時に AWS CloudHSM クラスターを指定し、変更することはできません。ただし、バックアップ履歴を共有するクラスターは、元のクラスターに置き換えることができます。これにより、必要に応じてクラスターを削除し、バックアップの 1 つから作成したクラス

ターに置き換えることができます。関連する AWS CloudHSM クラスターを完全に制御することにより、ユーザーとキーを管理し、HSM を作成および削除して、バックアップを使用および管理できます。

AWS CloudHSM キーストアを使用する準備ができたなら、関連付けられたクラスターに接続します AWS CloudHSM。いつでもカスタムキーストアを接続および切断できます。カスタムキーストアが接続されている場合は、その KMS キーを作成して使用できます。切断されると、AWS CloudHSM キーストアとその KMS キーを表示および管理できます。ただし、新しい KMS キーを作成したり、キーストアの KMS AWS CloudHSM キーを暗号化オペレーションに使用することはできません。

kmsuser Crypto User

ユーザーに代わって、関連付けられた AWS CloudHSM クラスターでキーマテリアルを作成および管理するために、`hsm` という名前のクラスター内の専用 AWS CloudHSM [Crypto User](#) (CU) AWS KMS を使用します `kmsuser`。 `kmsuser` CU は、クラスター内のすべての HSM に自動的に同期された標準の CU アカウントで、クラスターバッグに保存されます。

AWS CloudHSM キーストアを作成する前に、CloudHSM CLI [kmsuser のユーザー作成コマンド](#)を使用してクラスターに [CU アカウントを作成します](#)。AWS CloudHSM 次に、[AWS CloudHSM キーストアを作成する](#)ときに、`kmsuser` アカウントのパスワードを指定します AWS KMS。[カスタムキーストアを接続する](#)と、AWS KMS は CU `kmsuser` としてクラスターにログインし、パスワードをローテーションします。 `hsm` `kmsuser` パスワードを安全に保存する前に AWS KMS 暗号化します。パスワードがローテーションされる際、新しいパスワードは同じ方法で暗号化されて保存されます。

AWS KMS AWS CloudHSM キーストアが接続されている `kmsuser` 限り、 `hsm` ログインしたままになります。この CU アカウントは他の目的では使用しないでください。ただし、`kmsuser` CU アカウントの最終的な制御は保持されます。`kmsuser` が所有する [キーはいつでも検索することができます](#)。必要に応じて、[カスタムキーストアの切断](#)、`kmsuser` パスワードの変更、[kmsuser でクラスターへログイン](#)、および `kmsuser` が所有するキーの表示と管理が行えます。

`kmsuser` CU アカウントの作成手順については、「[kmsuser Crypto User を作成する](#)」を参照してください。

キーストアの KMS AWS CloudHSM キー

AWS KMS または AWS KMS API を使用して、AWS CloudHSM キーストア AWS KMS keys に作成できます。KMS キーで使用するのと同じ方法を使用します。唯一の違いは、AWS CloudHSM キーストアを識別し、キーマテリアルのオリジンが AWS CloudHSM クラスターであることを指定する必要があることです。

[AWS CloudHSM キーストアに KMS キーを作成する](#) AWS KMS と、 はに KMS キー AWS KMS を作成し、関連付けられたクラスターに 256 ビットの永続的でエクスポート不可能な Advanced Encryption Standard (AES) 対称キー材料を生成します。暗号化オペレーションで AWS KMS キーを使用すると、オペレーションは AWS CloudHSM クラスターベースの AES キーを使用してクラスターで実行されます。は異なるタイプの対称キーと非対称キー AWS CloudHSM をサポートしていますが、AWS CloudHSM キーストアは AES 対称暗号化キーのみをサポートします。

AWS KMS コンソールの キーストアで KMS AWS CloudHSM キーを表示し、コンソールオプションを使用してカスタムキーストア ID を表示できます。[DescribeKey](#) オペレーションを使用して、AWS CloudHSM キーストア ID と AWS CloudHSM クラスター ID を検索することもできます。

キーストアの KMS AWS CloudHSM キーは、 のどの KMS キーと同様に機能します AWS KMS。認可されたユーザーは、KMS キーの使用と管理のために同じアクセス許可が必要です。同じコンソールプロシージャと API オペレーションを使用して、キーストアの KMS AWS CloudHSM キーを表示および管理します。これには、KMS キーの有効化と無効化、タグとエイリアスの作成と使用、IAM ポリシーとキーポリシーの設定と変更が含まれます。AWS CloudHSM キーストアの KMS キーを暗号化オペレーションに使用し、カスタマーマネージドキーの使用をサポートする[統合 AWS サービス](#)で使用できます。ただし、[自動キーローテーション](#)を有効にしたり、キーストアの KMS キーに AWS CloudHSM キー[材料をインポート](#)したりすることはできません。

また、同じプロセスを使用して、キーストアの KMS AWS CloudHSM キーの[削除をスケジュール](#)します。待機期間が終了すると、 は KMS から KMS キー AWS KMS を削除します。次に、関連付けられた AWS CloudHSM クラスターから KMS キーのキー材料を削除するよう最善を尽くします。ただし、クラスターとそのバックアップから、手動で[孤立したキー材料を削除する](#)必要があります。

AWS CloudHSM キーストアへのアクセスを制御する

IAM ポリシーを使用して、AWS CloudHSM キーストアと AWS CloudHSM クラスターへのアクセスを制御します。キーポリシー、IAM ポリシー、および許可を使用して、AWS CloudHSM キーストア AWS KMS keys 内の へのアクセスを制御できます。ユーザー、グループ、およびロールには、実行する可能性が高いタスクに必要なアクセス許可のみを与えることをお勧めします。

AWS CloudHSM キーストアをサポートするには、 にクラスターに関する情報 AWS CloudHSM を取得するためのアクセス許可 AWS KMS が必要です。また、キーストアを AWS CloudHSM クラスターに接続する AWS CloudHSM ネットワークインフラストラクチャを作成するアクセス許可も必要です。これらのアクセス許可を取得するには、で `AWSServiceRoleForKeyManagementServiceCustomKeyStores` サービスにリンクされたロール AWS

KMS を作成します AWS アカウント。詳細については、「[AWS CloudHSM と Amazon EC2 リソースの管理 AWS KMS を承認する](#)」を参照してください。

AWS CloudHSM キーストアを設計するときは、キーストアを使用および管理しているプリンシパルに必要なアクセス許可のみを持っていることを確認してください。次のリストは、AWS CloudHSM キーストアマネージャーとユーザーに最低限必要なアクセス許可を示しています。

- AWS CloudHSM キーストアを作成および管理するプリンシパルには、AWS CloudHSM キーストア API オペレーションを使用するための次のアクセス許可が必要です。
 - `cloudhsm:DescribeClusters`
 - `kms:CreateCustomKeyStore`
 - `kms:ConnectCustomKeyStore`
 - `kms>DeleteCustomKeyStore`
 - `kms:DescribeCustomKeyStores`
 - `kms:DisconnectCustomKeyStore`
 - `kms:UpdateCustomKeyStore`
 - `iam:CreateServiceLinkedRole`
- キーストアに関連付けられている AWS CloudHSM AWS CloudHSM クラスターを作成および管理するプリンシパルには、AWS CloudHSM クラスターを作成および初期化するためのアクセス許可が必要です。これには、仮想プライベートクラウド (VPC) の作成または使用、サブネットの作成、Amazon EC2 インスタンスの作成権限が含まれます。また、HSM の作成と削除、およびバックアップの管理が必要な場合もあります。必要なアクセス権限のリストについては、「AWS CloudHSM ユーザーガイド」の「[AWS CloudHSM のアイデンティティとアクセス管理](#)」を参照してください。
- AWS CloudHSM キーストア AWS KMS keys で 作成および管理するプリンシパルには、で KMS キーを作成および管理するプリンシパルと同じ[アクセス許可](#)が必要です AWS KMS。 [キーストアの KMS キーのデフォルトキーポリシー](#)は、 の KMS キーのデフォルトキーポリシーと同じです AWS KMS。 AWS CloudHSM タグとエイリアスを使用して KMS キーへのアクセスを制御する[属性ベースのアクセスコントロール](#) (ABAC) は、AWS CloudHSM キーストアの KMS キーでも有効です。
- [暗号化オペレーション](#)に AWS CloudHSM キーストアの KMS キーを使用するプリンシパルには、[kms:Decrypt](#) などの KMS キーを使用して暗号化オペレーションを実行するアクセス許可が必要です。これらのアクセス許可は、キーポリシー、IAM ポリシーで付与できます。ただし、キーストアで KMS AWS CloudHSM キーを使用するための追加のアクセス許可は必要ありません。

AWS CloudHSM キーストアを作成する

アカウントに 1 つまたは複数の AWS CloudHSM キーストアを作成できます。各 AWS CloudHSM キーストアは、同じ AWS アカウント およびリージョン内の 1 つの AWS CloudHSM クラスターに関連付けられます。AWS CloudHSM キーストアを作成するときは、事前に[前提条件を構成する](#)必要があります。次に、AWS CloudHSM キーストアを使用する前に、そのキーストアを AWS CloudHSM クラスターに[接続](#)する必要があります。

メモ

KMS は IPv6 経由で AWS CloudHSM キーストアと通信できません。

既存の切断された AWS CloudHSM キーストアと同じプロパティ値をすべて含むキーストアを作成しようとすると、AWS KMS は新しい AWS CloudHSM キーストアを作成せず、例外をスローしたりエラーを表示したりしません。代わりに、重複を再試行の結果である可能性が高いものとして AWS KMS 認識し、既存の AWS CloudHSM キーストアの ID を返します。

AWS CloudHSM キーストアをすぐに接続する必要はありません。使用する準備ができるまで切断された状態にしておくことができます。ただし、正しく設定されていることを確認するために、[接続](#)して、[接続状態を表示](#)してから、[切断](#)するとよいかも知れません。

トピック

- [前提条件を構成する](#)
- [新しい AWS CloudHSM キーストアを作成する](#)

前提条件を構成する

各 AWS CloudHSM キーストアは AWS CloudHSM クラスターによってバックアップされます。AWS CloudHSM キーストアを作成するには、別のキーストアにまだ関連付けられていないアクティブな AWS CloudHSM クラスターを指定する必要があります。また、を使用してユーザーに代わってキーを作成および管理できる AWS KMS できるクラスターの HSMs に、専用の暗号化ユーザー (CU) を作成する必要があります。

AWS CloudHSM キーストアを作成する前に、次の操作を行います。

AWS CloudHSM クラスターを選択する

すべての AWS CloudHSM キーストアは、[1 つの AWS CloudHSM クラスターにのみ関連付け](#)られます。AWS CloudHSM キーストア AWS KMS keys で作成すると、は ID や Amazon リソースネーム (ARN) などの KMS キーメタデータ AWS KMS を作成します AWS KMS。その後、関連付けられたクラスターの HSM でキーマテリアルを作成します。[新しいクラスターを作成する AWS CloudHSM](#)ことも、既存のクラスターを使用することもできます。クラスターへの排他的アクセス AWS KMS は必要ありません。

選択した AWS CloudHSM クラスターは、AWS CloudHSM キーストアに永続的に関連付けられます。AWS CloudHSM キーストアを作成したら、関連付けられた[クラスターのクラスター ID を変更できます](#)が、指定するクラスターは元のクラスターとバックアップ履歴を共有する必要があります。無関係なクラスターを使用するには、新しい AWS CloudHSM キーストアを作成する必要があります。

選択する AWS CloudHSM クラスターには、次の特性が必要です。

- クラスターがアクティブである必要があります。

クラスターを作成して初期化し、プラットフォーム用の AWS CloudHSM クライアントソフトウェアをインストールしてから、クラスターをアクティブ化する必要があります。手順については、「AWS CloudHSM ユーザーガイド」の「[AWS CloudHSMの開始方法](#)」を参照してください。

- 相互 TLS (mTLS) が有効になっていません。

KMS はクラスターの mTLS をサポートしていません。この設定は有効にしないでください。

- クラスターは、キーストアと同じアカウントとリージョンに存在する必要があります。AWS CloudHSM あるリージョンの AWS CloudHSM キーストアを別のリージョンのクラスターに関連付けることはできません。複数のリージョンにキーインフラストラクチャを作成するには、各リージョンに AWS CloudHSM キーストアとクラスターを作成する必要があります。
- クラスターを同じアカウントおよびリージョン内の別のカスタムキーストアに関連付けることはできません。アカウントとリージョンの各 AWS CloudHSM キーストアは、異なる AWS CloudHSM クラスターに関連付ける必要があります。カスタムキーストアに関連付け済みのクラスターまたは関連付け済みのクラスターとバックアップ履歴を共有するクラスターを指定することはできません。バックアップ履歴を共有するクラスターには同じクラスター証明書があります。クラスターのクラスター証明書を表示するには、AWS CloudHSM コンソールまたは [DescribeClusters](#) オペレーションを使用します。

[AWS CloudHSM クラスターを別のリージョンにバックアップ](#)する場合、そのクラスターは別のクラスターと見なされ、そのリージョン内のカスタムキーストアにバックアップを関連付けることができます。ただし、2つのカスタムキーストアの KMS キーは、同じバックアップキーがあっても相互運用できません。はメタデータを暗号文に AWS KMS バインドするため、暗号化した KMS キーでのみ復号できます。

- クラスターは、リージョンの 2 つ以上のアベイラビリティーゾーンで[プライベートサブネット](#)を使用して設定する必要があります。AWS CloudHSM はすべてのアベイラビリティーゾーンでサポートされていないため、リージョン内のすべてのアベイラビリティーゾーンにプライベートサブネットを作成することをお勧めします。既存のクラスターのサブネットを再構成することはできませんが、クラスター構成の異なるサブネットを持つ[バックアップからクラスターを作成](#)することはできます。

 Important

AWS CloudHSM キーストアを作成したら、その AWS CloudHSM クラスター用に設定されたプライベートサブネットを削除しないでください。がクラスター設定内のすべてのサブネットを見つける AWS KMS ことができない場合、[カスタムキーストアへの接続](#)の試行はSUBNET_NOT_FOUND接続エラー状態で失敗します。詳細については、「[接続障害の修復方法](#)」を参照してください。

- [クラスターのセキュリティグループ](#) (cloudhsm-cluster-*<cluster-id>*-sg) には、ポート 2223-2225 で IPv4 経由の TCP トラフィックを許可するインバウンドルールとアウトバウンドルールを含める必要があります。インバウンドルールの Source とアウトバウンドルールの Destination は、セキュリティグループ ID と一致している必要があります。これらのルールは、クラスターの作成時にデフォルトで設定されます。変更または削除しないでください。
- クラスターは、異なるアベイラビリティーゾーンの少なくとも 2 つのアクティブな HSM を含む必要があります。HSMs の数を確認するには、AWS CloudHSM コンソールまたは [DescribeClusters](#) オペレーションを使用します。必要に応じて、[HSM を追加](#)できます。

信頼アンカー証明書を見つける

カスタムキーストアを作成するときは、AWS CloudHSM クラスターの信頼アンカー証明書をアップロードする必要があります AWS KMS。は AWS CloudHSM 、キーストアを関連する AWS CloudHSM クラスターに接続するために信頼アンカー証明書 AWS KMS を必要とします。

すべてのアクティブな AWS CloudHSM クラスターにはトラストアンカー証明書があります。[クラスターを初期化](#)する際、この証明書を生成し、customerCA.crt ファイルに保存して、クラスターに接続するホストにコピーしてください。

のkmsuser暗号化ユーザーを作成する AWS KMS

AWS CloudHSM キーストアを管理するには、選択したクラスターの [kmsuser Crypto User](#) (CU) アカウント AWS KMS にログインします。AWS CloudHSM キーストアを作成する前に、CU kmsuser を作成する必要があります。次に、AWS CloudHSM キーストアを作成するときに、 のパスワードを kmsuser に提供します AWS KMS。AWS CloudHSM キーストアを関連する AWS CloudHSM クラスターに接続するたびに、 は として AWS KMS ログインkmsuserし、kmsuserパスワードをローテーションします。

Important

kmsuser CU を作成するとき、2FA オプションを指定しないでください。その場合、AWS KMS はログインできず、AWS CloudHSM キーストアはこの AWS CloudHSM クラスターに接続できません。2FA を指定すると、元に戻すことはできません。代わりに、CU を削除して再作成する必要があります。

メモ

次の手順では、AWS CloudHSM クライアント SDK 5 コマンドラインツール [CloudHSM CLI](#) を使用します。CloudHSM CLI は、key-handle を key-reference に置き換えます。

2025 年 1 月 1 日、AWS CloudHSM はクライアント SDK 3 コマンドラインツール、CloudHSM 管理ユーティリティ (CMU)、およびキー管理ユーティリティ (KMU) のサポートを終了します。Client SDK 3 コマンドラインツールと Client SDK 5 コマンドラインツールの違いの詳細については、「AWS CloudHSM ユーザーガイド」の「[Client SDK 3 CMU および KMU から Client SDK 5 CloudHSM CLI への移行](#)」を参照してください。

1. 「AWS CloudHSM ユーザーガイド」の「[CloudHSM コマンドラインインターフェイス \(CLI\) の使用開始](#)」トピックに記載されている使用開始手順に従ってください。
2. [\[user create\]](#) コマンドを使用して、kmsuser という名前の CU を作成します。

パスワードは 7〜32 の英数字で構成する必要があります。大文字と小文字が区別され、特殊文字を含めることはできません。

次のコマンド例では、kmsuser CU を作成します。


```
aws-cloudhsm > user create --username kmsuser --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "kmsuser",
    "role": "crypto-user"
  }
}
```

新しい AWS CloudHSM キーストアを作成する

[前提条件を組み立て](#)た後、AWS KMS コンソールで、または [CreateCustomKeyStore](#) オペレーションを使用して、新しい AWS CloudHSM キーストアを作成できます。

AWS KMS コンソールの使用

で AWS CloudHSM キーストアを作成するときに AWS Management Console、ワークフローの一部として [前提条件](#) を追加および作成できます。ただし、プロセスは事前に構成しておくにより迅速になります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタムキーストア]、[AWS CloudHSM キーストア] の順に選択します。
4. [キーストアの作成] を選択します。
5. カスタムキーストアのわかりやすい名前を入力します。名前は、アカウント内のすべてのカスタムキーストアの間で、一意でなければなりません。

Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

6. AWS CloudHSM キーストアの [AWS CloudHSM クラスター](#) を選択します。または、新しい AWS CloudHSM クラスターを作成するには、クラスターの作成 AWS CloudHSM リンクを選択します。

メニューには、AWS CloudHSM キーストアにまだ関連付けられていないアカウントとリージョンの AWS CloudHSM クラスターが表示されます。クラスターは、カスタムキーストアとの関連付けの[要件を満たす](#)必要があります。

7. ファイルの選択を選択し、選択した AWS CloudHSM クラスターのトラストアンカー証明書をアップロードします。これは、[クラスターを初期化する](#)際に作成した customerCA.crt ファイルです。
8. 選択したクラスターで作成した [kmsuser Crypto User](#) (CU) のパスワードを入力します。
9. [作成] を選択します。

プロシージャが成功すると、新しい AWS CloudHSM キーストアがアカウントとリージョンの AWS CloudHSM キーストアのリストに表示されます。正常に完了しなかった場合は、問題を説明し、修正方法を示すエラーメッセージが表示されます。さらにヘルプが必要な場合は、「[カスタムキーストアのトラブルシューティング](#)」を参照してください。

既存の切断 AWS CloudHSM された AWS CloudHSM キーストアと同じプロパティ値をすべて含むキーストアを作成しようとする、AWS KMS は新しい AWS CloudHSM キーストアを作成しません。また、例外をスローしたり、エラーを表示したりしません。代わりに、は重複を再試行の結果である可能性が高いものとして AWS KMS 認識し、既存の AWS CloudHSM キーストアの ID を返します。

次へ: 新しい AWS CloudHSM キーストアは自動的に接続されません。AWS CloudHSM キーストア AWS KMS keys で 作成する前に、[カスタムキーストアを関連するクラスターに接続](#)する必要があります。AWS CloudHSM

AWS KMS API の使用

[CreateCustomKeyStore](#) オペレーションを使用して、アカウントとリージョンの AWS CloudHSM クラスターに関連付けられた新しい AWS CloudHSM キーストアを作成できます。これらの例では AWS Command Line Interface (AWS CLI) を使用しますが、サポートされている任意のプログラミング言語を使用できます。

CreateCustomKeyStore オペレーションでは、次のパラメータ値が必要です。

- CustomKeystorename — アカウント内で一意のカスタムキーストアのフレンドリ名。

⚠ Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

- CloudHsmClusterId – AWS CloudHSM キーストア [の要件を満たす](#) AWS CloudHSM クラスターのクラスター ID。
- KeyStorePassword — 指定したクラスター内の kmsuser CU アカウントのパスワード。
- TrustAnchorCertificate — [クラスターを初期化した](#)際に作成した customerCA.crt ファイルの内容。

次の例では、架空のクラスター ID を使用します。コマンドを実行する前に、有効なクラスター ID と置き換えます。

```
$ aws kms create-custom-key-store
  --custom-key-store-name ExampleCloudHSMKeyStore \
  --cloud-hsm-cluster-id cluster-1a23b4cdefg \
  --key-store-password kmsPswd \
  --trust-anchor-certificate <certificate-goes-here>
```

を使用している場合は AWS CLI、その内容の代わりにトラストアンカー証明書ファイルを指定できます。次の例では、customerCA.crt ファイルはルートディレクトリにあります。

```
$ aws kms create-custom-key-store
  --custom-key-store-name ExampleCloudHSMKeyStore \
  --cloud-hsm-cluster-id cluster-1a23b4cdefg \
  --key-store-password kmsPswd \
  --trust-anchor-certificate file://customerCA.crt
```

オペレーションが正常に終了したら、次のレスポンス例に示すように、CreateCustomKeyStore はカスタムキーストア ID を返します。

```
{
  "CustomKeyStoreId": cks-1234567890abcdef0
}
```

オペレーションが失敗した場合は、例外で示されているエラーを修正して、もう一度試してください。その他のヘルプについては、「[カスタムキーストアのトラブルシューティング](#)」を参照してください。

既存の切断 AWS CloudHSM された AWS CloudHSM キーストアと同じプロパティ値をすべて含むキーストアを作成しようとする、AWS KMS は新しい AWS CloudHSM キーストアを作成せず、例外をスローしたり、エラーを表示したりしません。代わりに、は重複を再試行の結果である可能性が高いものとして AWS KMS 認識し、既存の AWS CloudHSM キーストアの ID を返します。

次へ: AWS CloudHSM キーストアを使用するには、[そのキーストアを AWS CloudHSM クラスターに接続します](#)。

AWS CloudHSM キーストアを表示する

AWS KMS コンソールまたは [DescribeCustomKeyStores](#) オペレーションを使用して、各アカウントとリージョンの AWS CloudHSM キーストアを表示できます。

AWS KMS コンソールの使用

で AWS CloudHSM キーストアを表示すると AWS Management Console、以下が表示されます。

- カスタムキーストアの名前と ID
- 関連付けられた AWS CloudHSM クラスターの ID
- クラスター内の HSM の数
- 現在の接続ステータス

Disconnected の接続状態 (ステータス) の値は、カスタムキーストアが新しく、一度も接続されていないか、[AWS CloudHSM クラスターから意図的に切断](#)されたことを示します。ただし、接続されたカスタムキーストアで KMS キーを使用しようとする場合、カスタムキーストアまたはその AWS CloudHSM クラスターに問題がある可能性があります。ヘルプについては、「[失敗した KMS キーを修正するには](#)」を参照してください。

特定のアカウントとリージョンの AWS CloudHSM キーストアを表示するには、次の手順を使用します。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクタを使用します。

3. ナビゲーションペインで、[カスタムキーストア]、[AWS CloudHSM キーストア] の順に選択します。

表示をカスタマイズするには、[Create key store (キーストアを作成)] ボタンの下に表示される歯車アイコンをクリックします。

AWS KMS API の使用

AWS CloudHSM キーストアを表示するには、[DescribeCustomKeyStores](#) オペレーションを使用します。デフォルトでは、このオペレーションは、アカウントとリージョンのすべてのカスタムキーストアを返します。ただし、CustomKeyId または CustomKeyName パラメータのどちらかを使用して (両方は使用できません) 出力を特定のカスタムキーストアに制限できます。AWS CloudHSM キーストアの場合、出力はカスタムキーストア ID と名前、カスタムキーストアタイプ、関連付けられた AWS CloudHSM クラスターの ID、および接続状態で構成されます。接続状態はエラーを示す場合、出力にエラーの理由を説明するエラーコードも含まれています。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

たとえば、次のコマンドは、アカウントとリージョンのすべてのカスタムキーストアを返します。Limit パラメータと Marker パラメータを使用して、出力のカスタムキーストアをページ分割できます。

```
$ aws kms describe-custom-key-stores
```

次のコマンド例では、CustomKeyName パラメータを使用して ExampleCloudHSMKeyStore というフレンドリ名の唯一のカスタムキーストアを取得します。各コマンドで CustomKeyName パラメータまたは CustomKeyId パラメータのどちらかを使用できますが、両方を使用することはできません。

次の出力例は、クラスター AWS CloudHSM に接続されている AWS CloudHSM キーストアを表します。

Note

キーストアと外部 AWS CloudHSM キーストアを区別するために、CustomKeyType フィールドが DescribeCustomKeyStores レスポンスに追加されました。

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleCloudHSMKeyStore
{
  "CustomKeyStores": [
    {
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionState": "CONNECTED",
      "CreationDate": "1.499288695918E9",
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate appears here>"
    }
  ]
}
```

ConnectionState の Disconnected は、カスタムキーストアが接続されたことがないか、[AWS CloudHSM クラスタから意図的に切断](#)されたことを示します。ただし、接続されたキーストアで KMS AWS CloudHSM キーを使用しようとする場合、AWS CloudHSM キーストアまたはその AWS CloudHSM クラスタに問題がある可能性があります。ヘルプについては、「[失敗した KMS キーを修正するには](#)」を参照してください。

カスタムキーストアの ConnectionState が FAILED である場合、DescribeCustomKeyStores レスポンスには、エラーの理由を説明する ConnectionErrorCode 要素が含まれています。

たとえば、次の出力では、INVALID_CREDENTIALS 値は、[kmsuser パスワードが無効である](#)ために、カスタムキーストアの接続に失敗したことを示しています。このエラーやその他の接続エラーに関するヘルプについては、「」を参照してください [カスタムキーストアのトラブルシューティング](#)。

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
  "CustomKeyStores": [
    {
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionErrorCode": "INVALID_CREDENTIALS",
      "ConnectionState": "FAILED",
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "CreationDate": "1.499288695918E9",
      "TrustAnchorCertificate": "<certificate appears here>"
    }
  ]
}
```

```
}  
]  
}
```

詳細はこちら:

- [外部キーストアを表示する](#)
- [キーストア内の KMS AWS CloudHSM キーを特定する](#)
- [を使用した AWS KMS API コールのログ記録 AWS CloudTrail](#)

AWS CloudHSM キーストア設定を編集する

既存の AWS CloudHSM キーストアの設定を変更できます。カスタムキーストアは AWS CloudHSM 、 クラスターを切断する必要があります。

AWS CloudHSM キーストア設定を編集するには :

1. カスタムキーストアの AWS CloudHSM クラスターから、[カスタムキーストアを切断します](#)。

カスタムキーストアが切断されている間は、カスタムキーストアで AWS KMS keys (KMS キー) を作成することはできません。また、[暗号化オペレーション](#)に格納されている KMS キーを使用することはできません。

2. 1 つ以上の AWS CloudHSM キーストア設定を編集します。

カスタムキーストアで次の設定を編集できます。

カスタムキーストアのわかりやすい名前。

新しい分かりやすい名前を入力します。新しい名前は、 内のすべてのカスタムキーストアで一意である必要があります AWS アカウント。

Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

関連付けられたクラスターの AWS CloudHSM クラスター ID。

この値を編集して、関連する AWS CloudHSM クラスターを元のクラスターに置き換えます。この機能を使用して、AWS CloudHSM クラスターが破損したり削除されたりした場合にカスタムキーストアを修復できます。

バックアップ履歴を元の AWS CloudHSM クラスターと共有し、異なるアベイラビリティーゾーン内の 2 つのアクティブな HSMs を含むカスタムキーストアとの関連付けの要件を満たす クラスターを指定します。バックアップ履歴を共有するクラスターには同じクラスター証明書があります。クラスターのクラスター証明書を表示するには、[DescribeClusters](#) オペレーションを使用します。編集機能を使用してカスタムキーストアを無関係な AWS CloudHSM クラスターと関連付けることはできません。

[kmsuser Crypto User](#) (CU) の現在のパスワード。

クラスター内の CU kmsuser の現在のパスワード AWS KMS を指示します AWS CloudHSM。このアクションでは、AWS CloudHSM クラスター内の CU kmsuser のパスワードは変更されません。

AWS CloudHSM クラスター内の CU kmsuser のパスワードを変更する場合は、この機能を使用して新しい kmsuser パスワードを に伝え AWS KMS ます。それ以外の場合、AWS KMS はクラスターにログインできず、カスタムキーストアをクラスターに接続しようとするすべての試行は失敗します。

3. [カスタムキーストアを AWS CloudHSM クラスターに再接続します。](#)

キーストア設定を編集する

AWS CloudHSM キーストアの設定は、AWS KMS コンソールで編集するか、[UpdateCustomKeyStore](#) オペレーションを使用して編集できます。

AWS KMS コンソールの使用

AWS CloudHSM キーストアを編集するときに、設定可能な値のいずれかまたは を変更できます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクタを使用します。

3. ナビゲーションペインで、[カスタムキーストア]、[AWS CloudHSM キーストア] の順に選択します。
4. 編集する AWS CloudHSM キーストアの行を選択します。

[接続の状態] 列の値が [切断済み] になっていない場合は、カスタムキーストアを切断してから編集する必要があります。([Key store actions] (キーストアアクション) メニューから [Disconnect] (切断) を選択します)。

AWS CloudHSM キーストアが切断されている間は、AWS CloudHSM キーストアとその KMS キーを管理できますが、キーストアで KMS AWS CloudHSM キーを作成または使用することはできません。

5. [Key store actions] (キーストアアクション) メニューから [Edit] (編集) を選択します。
6. 次のいずれかのアクションを実行します。
 - カスタムキーストアの新しいわかりやすい名前を入力します。
 - 関連するクラスターの AWS CloudHSM クラスター ID を入力します。
 - 関連付けられた AWS CloudHSM クラスター内の kmsuser Crypto User の現在のパスワードを入力します。
7. [Save] を選択します。

プロシージャが正常に完了すると、編集した設定を説明するメッセージが表示されます。正常に行われなかった場合は、問題を説明し、修正方法を示すエラーメッセージが表示されます。さらにヘルプが必要な場合は、「[カスタムキーストアのトラブルシューティング](#)」を参照してください。

8. [カスタムキーストアを再接続します。](#)

AWS CloudHSM キーストアを使用するには、編集後に再接続する必要があります。AWS CloudHSM キーストアは切断されたままにすることができます。ただし、切断されている間は、AWS CloudHSM キーストアで KMS キーを作成したり、[暗号化オペレーション](#)でキーストアで KMS AWS CloudHSM キーを使用したりすることはできません。

AWS KMS API の使用

AWS CloudHSM キーストアのプロパティを変更するには、[UpdateCustomKeyStore](#) オペレーションを使用します。同じコマンドで、カスタムキーストアの数個のプロパティを変更できます。オペレーションが成功すると、は HTTP 200 レスポンスとプロパティのない JSON オブジェクト AWS

KMS を返します。変更が有効になっていることを確認するには、[DescribeCustomKeyStores](#) オペレーションを使用します。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

まず、[DisconnectCustomKeyStore](#) を使用して [カスタムキーストアをクラスターから切断](#) します。AWS CloudHSM 例のカスタムキーストア ID `cks-1234567890abcdef0` を実際の ID に置き換えます。

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

最初の例では、[UpdateCustomKeyStore](#) を使用して AWS CloudHSM キーストアのフレンドリ名を変更します `DevelopmentKeys`。コマンドは、`CustomKeyId` パラメータを使用して AWS CloudHSM キーストアを識別し、`CustomKeyName` を使用してカスタムキーストアの新しい名前を指定します。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --new-custom-key-store-name DevelopmentKeys
```

次の例では、AWS CloudHSM キーストアに関連付けられているクラスターを、同じクラスターの別のバックアップに変更します。コマンドは、`CustomKeyId` パラメータを使用して AWS CloudHSM キーストアを識別し、`CloudHsmClusterId` パラメータを使用して新しいクラスター ID を指定します。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --cloud-hsm-cluster-id cluster-1a23b4cdefg
```

次の例では、現在の `kmsuser` パスワードが AWS KMS であることを指示します `ExamplePassword`。コマンドは、`CustomKeyId` パラメータを使用して AWS CloudHSM キーストアを識別し、`KeyStorePassword` パラメータを使用して現在のパスワードを指定します。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --key-store-password ExamplePassword
```

最後のコマンドは、AWS CloudHSM キーストアを AWS CloudHSM クラスターに再接続します。カスタムキーストアは切断された状態のままにできますが、新しい KMS キーを作成したり、[暗号化オペレーション](#) で既存の KMS キーを使用したりする前に接続する必要があります。カスタムキーストア ID 例を実際の ID と置き換えます。

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

AWS CloudHSM キーストアを接続する

新しい AWS CloudHSM キーストアが接続されていません。AWS CloudHSM キーストア AWS KMS keys で作成して使用する前に、関連付けられた AWS CloudHSM クラスターに接続する必要があります。AWS CloudHSM キーストアはいつでも接続および切断でき、[その接続状態を表示できます](#)。

AWS CloudHSM キーストアを接続する必要はありません。AWS CloudHSM キーストアは無期限に切断状態のままにして、使用する必要がある場合にのみ接続できます。ただし、定期的に接続をテストして、接続が正しく接続できることを確認するとよいでしょう。

Note

AWS CloudHSM キーストアは、キーストアが接続されていないか、明示的に切断した場合にのみ接続DISCONNECTED状態になります。AWS CloudHSM キーストアの接続状態がCONNECTEDであっても、使用できない場合は、関連付けられた AWS CloudHSM クラスターがアクティブで、少なくとも1つのアクティブな HSMs が含まれていることを確認してください。接続障害については、[the section called “カスタムキーストアのトラブルシューティング”](#) を参照してください。

AWS CloudHSM キーストアを接続すると、は関連付けられた AWS CloudHSM クラスター AWS KMS を見つけて接続し、[kmsuserCrypto User](#) (CU) として AWS CloudHSM クライアントにログインしてから、AWS CloudHSM キーストアが接続されている限り、AWS CloudHSM クライアントにログインした kmsuser password. AWS KMS remains をローテーションします。

接続を確立するために、はクラスターの Virtual Private Cloud (VPC) kms-**<custom key store ID>**に という名前[のセキュリティグループ](#) AWS KMS を作成します。セキュリティグループには、クラスターセキュリティグループからのインバウンドトラフィックを許可する単一のルールがあります。AWS KMS また、は、クラスターのプライベートサブネットの各アベイラビリティゾーンに[Elastic Network Interface](#) (ENI) を作成します。は、ENIsをクラスターkms-**<cluster ID>**のセキュリティグループとセキュリティグループ AWS KMS に追加します。各 ENI の説明は KMS managed ENI for cluster **<cluster-ID>** です。

接続プロセスは、完了するまでに長い時間 (最長 20 分) がかかる場合があります。

AWS CloudHSM キーストアを接続する前に、キーストアが要件を満たしていることを確認します。

- 関連付けられた AWS CloudHSM クラスターには、少なくとも 1 つのアクティブな HSM が含まれている必要があります。クラスター内の HSMs の数を確認するには、AWS CloudHSM コンソールでクラスターを表示するか、[DescribeClusters](#) オペレーションを使用します。必要に応じて、[HSM を追加](#)できます。
- クラスターには [kmsuserCrypto User](#) (CU) アカウントが必要ですが、AWS CloudHSM キーストアを接続するときにその CU をクラスターにログインすることはできません。ログアウトのヘルプについては、「[ログアウトして再接続する方法](#)」を参照してください。
- AWS CloudHSM キーストアの接続状態を DISCONNECTING または にすることはできません FAILED。接続状態を表示するには、AWS KMS コンソールまたは [DescribeCustomKeyStores](#) レスポンスを使用します。接続ステータスが FAILED の場合、カスタムキーストアを切断し、問題を解決してから接続します。

接続障害については、[接続障害の修復方法](#) を参照してください。

AWS CloudHSM キーストアが接続されたら、[そこに KMS キーを作成し](#)、[暗号化オペレーション](#)で既存の KMS キーを使用できます。

AWS CloudHSM キーストアに接続して再接続する

AWS KMS コンソールで、または [ConnectCustomKeyStore](#) オペレーションを使用して、AWS CloudHSM キーストアに接続または再接続できます。

AWS KMS コンソールの使用

で AWS CloudHSM キーストアを接続するには AWS Management Console、カスタム AWS CloudHSM キーストアページからキーストアを選択します。この接続処理には、完了までに最大で 20 分かかります。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクタを使用します。
3. ナビゲーションペインで、[カスタムキーストア]、[AWS CloudHSM キーストア] の順に選択します。
4. 接続する AWS CloudHSM キーストアの行を選択します。

AWS CloudHSM キーストアの接続状態が失敗の場合は、接続する前に[カスタムキーストアを切断](#)する必要があります。

5. [Key store actions] (キーストアアクション) メニューから [Connect] (接続) を選択します。

AWS KMS は、カスタムキーストアの接続プロセスを開始します。関連付け済みの AWS CloudHSM クラスターを見つけ、必要なネットワークインフラストラクチャを構築し、そのインフラストラクチャに接続し、AWS CloudHSM クラスターに kmsuser CU としてログインして、kmsuser パスワードをローテーションします。操作が完了すると、接続状態が [接続済み] に変わります。

オペレーションが失敗すると、失敗の理由を説明するエラーメッセージが表示されます。再度接続を試みる前に、AWS CloudHSM キーストア [の接続状態を確認してください](#)。状態が [失敗] になっている場合は、[カスタムキーストアを切断](#)してからキーストアをもう一度接続する必要があります。ヘルプが必要な場合は、「[カスタムキーストアのトラブルシューティング](#)」を参照してください。

次の手順: [the section called “キーストアに KMS AWS CloudHSM キーを作成する”](#)

AWS KMS API の使用

切断された AWS CloudHSM キーストアを接続するには、[ConnectCustomKeyStore](#) オペレーションを使用します。関連付けられた AWS CloudHSM クラスターには少なくとも 1 つのアクティブな HSM が含まれている必要があり、接続状態を にすることはできません FAILED。

接続プロセスは、完了するまでに長い時間 (最長 20 分) かかります。すぐに失敗しない限り、オペレーションは HTTP 200 レスポンスとプロパティを含まない JSON オブジェクトを返します。ただし、この初期レスポンスは接続に成功したことを示していません。カスタムキーストアの接続ステータスを判断するときは、[DescribeCustomKeyStores](#) のレスポンスを確認します。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

AWS CloudHSM キーストアを識別するには、カスタムキーストア ID を使用します。ID はコンソールの [Custom key stores] (カスタムキーストア) ページで確認できます。またはパラメータなしで [DescribeCustomKeyStores](#) オペレーションを使って確認することもできます。この例を実行する前に、例の ID を有効な ID に置き換えます。

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

AWS CloudHSM キーストアが接続されていることを確認するには、[DescribeCustomKeyStores](#) オペレーションを使用します。デフォルトでは、このオペレーションは、アカウントとリージョンのすべてのカスタムキーストアを返します。ただし、CustomKeyId または CustomKeyName パラメータのどちらかを使用して (両方は使用できません) レスポンスを特定のカスタムキースト

アに制限できます。CONNECTED の ConnectionState 値は、カスタムキーストアがその AWS CloudHSM クラスターに接続されていることを示します。

Note

キーストアと外部 AWS CloudHSM キーストアを区別するために、CustomKeyStoreType フィールドが DescribeCustomKeyStores レスポンスに追加されました。

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
  "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleCloudHSMKeyStore",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "CONNECTED"
    }
  ],
}
```

ConnectionState 値が [FAILED] の場合、ConnectionErrorCode 要素が失敗の原因を示します。この場合、AWS CloudHSM クラスター ID のアカウントでクラスター AWS KMS が見つかりませんでした cluster-1a23b4cdefg。クラスターを削除した場合、元のクラスターの [バックアップから復元する](#)ことができ、その後でカスタムキーストアの [クラスター ID を編集](#)できます。接続エラーコードの対処方法については「[接続障害の修復方法](#)」を参照してください。

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
  "CustomKeyStores": [
    {
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "CustomKeyStoreType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "FAILED",
      "ConnectionErrorCode": "CLUSTER_NOT_FOUND"
    }
  ],
}
```



```
}
```

AWS CloudHSM キーストアを切断する

AWS CloudHSM キーストアを切断すると、は AWS CloudHSM クライアントから AWS KMS ログアウトし、関連付けられた AWS CloudHSM クラスターから切断して、接続をサポートするために作成したネットワークインフラストラクチャを削除します。

AWS CloudHSM キーストアが切断されている間は、AWS CloudHSM キーストアとその KMS キーを管理できますが、キーストアで KMS AWS CloudHSM キーを作成または使用することはできません。PendingDeletion でない限り、キーストアの接続ステータスは DISCONNECTED で、カスタムキーストアの KMS キーの [キーステータス](#) は Unavailable です。AWS CloudHSM キーストアはいつでも再接続できます。

Note

AWS CloudHSM キーストアは、キーストアが接続されていないか、明示的に切断した場合にのみ接続DISCONNECTED状態になります。AWS CloudHSM キーストアの接続状態が CONNECTED であっても、使用できない場合は、関連付けられた AWS CloudHSM クラスターがアクティブで、少なくとも 1 つのアクティブな HSMs が含まれていることを確認してください。接続障害については、[the section called “カスタムキーストアのトラブルシューティング”](#) を参照してください。

カスタムキーストアを切断すると、そのキーストアの KMS キーはただちに使用できなくなります (結果整合性の影響を受ける)。ただし、KMS キーで保護された [データキー](#) により暗号化されたリソースは、KMS キーがデータキーの復号化などで再び使用されるまでは影響を受けません。この問題は AWS のサービスに影響します。その多くが、リソースを保護するためにデータキーを使用しています。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

Note

カスタムキーストアが切断されている間は、カスタムキーストアで KMS キーを作成したり、暗号化オペレーションで既存の KMS キーを使用したりする試みはすべて失敗します。このオペレーションにより、ユーザーが機密データを保存したりアクセスしたりすることを防ぐことができます。

カスタムキーストアの切断の影響をより正確に推定するには、カスタムキーストアで [KMS キーを識別し、その過去の使用状況を特定](#) します。

AWS CloudHSM キーストアは、次のような理由で切断できます。

- パスワードをローテーション **kmsuser** する。AWS KMS は AWS CloudHSM クラスターに接続するたびに kmsuser パスワードを変更します。パスワードローテーションを強制的に実行するには、切断して再接続します。
- AWS CloudHSM クラスター内の KMS キーのキーマテリアルを監査します。カスタムキーストアを切断すると、は AWS CloudHSM クライアントの [kmsuser Crypto User](#) Account から AWS KMS ログアウトします。これにより、クラスターに kmsuser CU としてログインし、KMS キーのキーマテリアルを監査および管理することができます。
- AWS CloudHSM キーストアですべての KMS キーをただちに無効にするには または [DisableKey](#) オペレーションを使用して、キーストアで [KMS キーを無効化および再有効化](#) できます。AWS CloudHSM AWS Management Console これらのオペレーションは迅速に完了しますが、一度に処理される KMS キーは 1 つです。AWS CloudHSM キーストアを切断すると、キーストア内のすべての KMS キーのキーステータスが AWS CloudHSM すぐに変更され Unavailable、暗号化オペレーションで使用されなくなります。
- 失敗した接続試行を修復するには。AWS CloudHSM キーストアの接続に失敗した場合 (カスタムキーストアの接続状態が の場合 FAILED)、再度接続を試みる前に AWS CloudHSM キーストアを切断する必要があります。

AWS CloudHSM キーストアを切断する

AWS KMS コンソールで、または [DisconnectCustomKeyStore](#) オペレーションを使用して、AWS CloudHSM キーストアを切断できます。

AWS KMS コンソールを使用して切断する

AWS KMS コンソールで接続された AWS CloudHSM キーストアを切断するには、カスタム AWS CloudHSM キーストアページからキーストアを選択します。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[カスタムキーストア]、[AWS CloudHSM キーストア] の順に選択します。

4. 切断する外部キーストアの行を選択します。
5. [Key store actions] (キーストアアクション) メニューから [Disconnect] (切断) を選択します。

操作が完了すると、接続状態が [切断] から [切断済み] に変わります。オペレーションが失敗した場合は、問題を説明し、修正方法を示すエラーメッセージが表示されます。さらにヘルプが必要な場合は、「[カスタムキーストアのトラブルシューティング](#)」を参照してください。

AWS KMS API を使用して切断する

接続された AWS CloudHSM キーストアを切断するには、[DisconnectCustomKeyStore](#) オペレーションを使用します。オペレーションが成功すると、は HTTP 200 レスポンスとプロパティのない JSON オブジェクト AWS KMS を返します。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

この例では、AWS CloudHSM キーストアを切断します。この例を実行する前に、例の ID を有効な ID に置き換えます。

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

AWS CloudHSM キーストアが切断されていることを確認するには、[DescribeCustomKeyStores](#) オペレーションを使用します。デフォルトでは、このオペレーションは、アカウントとリージョンのすべてのカスタムキーストアを返します。ただし、CustomKeyId または CustomKeyName パラメータ のどちらかを使用して (両方は使用できません) レスポンスを特定のカスタムキーストアに制限できます。のConnectionState値は、このサンプル AWS CloudHSM キーストアが AWS CloudHSM クラスターに接続されていないDISCONNECTEDことを示します。

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
  "CustomKeyStores": [
    {
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionState": "DISCONNECTED",
      "CreationDate": "1.499288695918E9",
      "CustomKeyId": "cks-1234567890abcdef0",
      "CustomKeyName": "ExampleKeyStore",
      "CustomKeyType": "AWS_CLOUDHSM",
      "TrustAnchorCertificate": "<certificate string appears here>"
    }
  ],
```

```
}
```

AWS CloudHSM キーストアを削除する

AWS CloudHSM キーストアを削除すると、AWS KMS は、AWS CloudHSM クラスターとの関連付けに関する情報など、AWS CloudHSM キーストアに関するすべてのメタデータを KMS から削除します。このオペレーションは、AWS CloudHSM クラスター、その HSMs、またはそのユーザーには影響しません。同じ AWS CloudHSM クラスターに関連付けられている新しい AWS CloudHSM キーストアを作成できますが、削除オペレーションを元に戻すことはできません。

削除できるのは、AWS CloudHSM クラスターから切断され、が含まれていない AWS CloudHSM キーストアのみです AWS KMS keys。カスタムキーストアを削除する前に、次の手順を実行します。

- いずれの[暗号化オペレーション](#)でも、キーストア内で KMS キーを一切使用する必要がないことを確認します。次に、キーストアからのすべての KMS キーの[削除をスケジュール](#)します。キーストアで KMS AWS CloudHSM キーを検索する方法については、「」を参照してください[キーストアで KMS AWS CloudHSM キーを検索する](#)。
- すべての KMS キーが削除されたことを確認します。キーストアの KMS AWS CloudHSM キーを表示するには、「」を参照してください[the section called “キーストア内の KMS AWS CloudHSM キーを特定する”](#)。
- クラスターから[AWS CloudHSM キーストアを切断します](#)。AWS CloudHSM

AWS CloudHSM キーストアを削除する代わりに、関連付けられた AWS CloudHSM クラスターからキーストアを[切断](#)することを検討してください。AWS CloudHSM キーストアが切断されている間は、AWS CloudHSM キーストアとその を管理できます AWS KMS keys。ただし、キーストアで KMS AWS CloudHSM キーを作成または使用することはできません。AWS CloudHSM キーストアはいつでも再接続できます。

AWS CloudHSM キーストアを削除する

AWS CloudHSM キーストアは、AWS KMS コンソールで、または [DeleteCustomKeyStore](#) オペレーションを使用して削除できます。

AWS KMS コンソールの使用

で AWS CloudHSM キーストアを削除するには AWS Management Console、カスタム AWS CloudHSM キーストアページからキーストアを選択します。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[カスタムキーストア]、[AWS CloudHSM キーストア] の順に選択します。
4. 削除する AWS CloudHSM キーストアを表す行を見つけます。AWS CloudHSM キーストアの接続状態が切断されていない場合は、削除する前に[AWS CloudHSM キーストアを切断](#)する必要があります。
5. [Key store actions] (キーストアアクション) メニューから [Delete] (削除) を選択します。

オペレーションが完了すると、成功メッセージが表示され、AWS CloudHSM キーストアはキーストアリストに表示されなくなります。オペレーションが正常に行われなかった場合、問題を説明し、修正方法を示すエラーメッセージが表示されます。さらにヘルプが必要な場合は、「[カスタムキーストアのトラブルシューティング](#)」を参照してください。

AWS KMS API の使用

AWS CloudHSM キーストアを削除するには、[DeleteCustomKeyStore](#) オペレーションを使用します。オペレーションが成功すると、は HTTP 200 レスポンスとプロパティのない JSON オブジェクト AWS KMS を返します。

開始するには、AWS CloudHSM キーストアに が含まれていないことを確認します AWS KMS keys。KMS キーが含まれているカスタムキーストアを削除することはできません。最初のコマンド例では、[ListKeys](#) と [DescribeKey](#) を使用して、サンプル `cks-1234567890abcdef0` カスタム AWS CloudHSM キーストア ID を持つキーストア AWS KMS keys で を検索します。この場合、コマンドは KMS キーを返しません。KMS キーが存在する場合は、[ScheduleKeyDeletion](#) オペレーションを使用して、各 KMS キーの削除をスケジュールします。

Bash

```
for key in $(aws kms list-keys --query 'Keys[*].KeyId' --output text) ;  
do aws kms describe-key --key-id $key |  
grep '"CustomKeyIdStoreId": "cks-1234567890abcdef0"' --context 100; done
```

PowerShell

```
PS C:\> Get-KMSKeyList | Get-KMSKey | where CustomKeyIdStoreId -eq  
'cks-1234567890abcdef0'
```

次に、AWS CloudHSM キーストアを切断します。このコマンド例では、[DisconnectCustomKeyStore](#) オペレーションを使用して、AWS CloudHSM キーストアを AWS CloudHSM クラスターから切断します。このコマンドを実行する前に、例のカスタムキーストア ID を有効な ID に置き換えます。

Bash

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

PowerShell

```
PS C:\> Disconnect-KMSCustomKeyStore -CustomKeyStoreId cks-1234567890abcdef0
```

カスタムキーストアが切断された後、[DeleteCustomKeyStore](#) オペレーションを使用して、[カスタムキーストア](#) を削除できます。

Bash

```
$ aws kms delete-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

PowerShell

```
PS C:\> Remove-KMSCustomKeyStore -CustomKeyStoreId cks-1234567890abcdef0
```

カスタムキーストアのトラブルシューティング

AWS CloudHSM キーストアは、利用可能で回復力があるように設計されています。ただし、AWS CloudHSM キーストアを運用し続けるために修復する必要があるエラー条件がいくつかあります。

トピック

- [使用できない KMS キーを修正するには](#)
- [失敗した KMS キーを修正するには](#)
- [接続障害の修復方法](#)
- [暗号化オペレーションの失敗に対応するには](#)
- [無効な kmsuser 認証情報の修正方法](#)

- [孤立したキーマテリアルを削除する方法](#)
- [KMS キーの削除されたキーマテリアルを復旧するには](#)
- [kmsuser としてログインする方法](#)

使用できない KMS キーを修正するには

[キーストアのキー状態](#)は通常 `Enabled` です。AWS KMS keys AWS CloudHSM すべての KMS キーと同様に、キーストアで KMS キーを無効にするか、削除をスケジュールすると、AWS CloudHSM キーの状態が変わります。ただし、他の KMS キーとは異なり、カスタムキーストアの KMS キーには、`Unavailable` の[キーステータス](#)もあります。

`Unavailable` のキーステータスには、KMS キーが、意図的に[切断](#)されたカスタムキーストアの内部に存在し、再接続の試み (実行している場合) に失敗したことを示しています。KMS キーは使用できませんが、KMS キーを表示および管理することはできます。ただし、[暗号化オペレーション](#)で使用することはできません。

KMS キーのキーステータスを確認するには、[Customer managed keys] (カスタマーマネージドキー) ページで、KMS キーの [Status] (ステータス) フィールドを表示します。または、[DescribeKey](#) オペレーションを使用して、レスポンスの `KeyState` 要素を表示します。詳細については、「[キーの特定と表示](#)」を参照してください。

切断されたカスタムキーストアの KMS キーは、`Unavailable` または `PendingDeletion` のキーステータスとなります。カスタムキーストアが切断されている場合でも、カスタムキーストアからの削除がスケジュールされている KMS キーのキーステータスは `Pending Deletion` になります。これにより、カスタムキーストアに再接続することなく、スケジュールされたキーの削除をキャンセルできます。

使用できない KMS キーを修正するには、[カスタムキーストアを再接続します](#)。カスタムキーストアを再接続すると、カスタムキーストア内の KMS キーのキーステータスは、`Enabled` や `Disabled` などの以前のステータスに自動的に復元されます。削除保留中の KMS キーは `PendingDeletion` ステータスのままです。ただし、問題が解決しない間は、[使用できない KMS キーを有効および無効にしても](#)、キーステータスに変更されません。有効または無効のアクションは、キーが使用可能になったときにのみ適用されます。

接続の失敗に関するヘルプについては、「[接続障害の修復方法](#)」を参照してください。

失敗した KMS キーを修正するには

AWS CloudHSM キーストアでの KMS キーの作成と使用に関する問題は、AWS CloudHSM キーストア、関連する AWS CloudHSM クラスター、KMS キー、またはそのキーマテリアルの問題が原因で発生する可能性があります。

AWS CloudHSM キーストアが AWS CloudHSM クラスターから切断されると、カスタムキーストアの KMS キーのキー状態は `Unavailable` になります。切断されたキーストアで KMS AWS CloudHSM キーを作成するすべてのリクエストは `CustomKeyStoreInvalidStateException` 例外を返します。データキーを暗号化、復号、再暗号化、または生成するすべてのリクエストは、`KMSInvalidStateException` 例外を返します。問題を解決するには、[AWS CloudHSM キーストアを再接続します](#)。

ただし、[暗号化オペレーション](#)のために AWS CloudHSM キーストアで KMS キーを使用しようとする `Enabled` と、キーの状態が `Unavailable` で AWS CloudHSM、キーストアの接続状態が `Disconnected` であっても失敗する可能性があります `Connected`。これは、以下のいずれかの条件によって発生する可能性があります。

- KMS キーのキーマテリアルが、関連付けられた AWS CloudHSM クラスターから削除された可能性があります。調査するには、KMS キーのキーマテリアルの [キー ID を探し](#)、必要に応じて [キーマテリアルの復旧](#)を試みます。
- AWS CloudHSM キーストアに関連付けられている AWS CloudHSM クラスターからすべての HSMs が削除されました。暗号化オペレーションで キーストアで KMS AWS CloudHSM キーを使用するには、その AWS CloudHSM クラスターに少なくとも 1 つのアクティブな HSM が含まれている必要があります。AWS CloudHSM クラスター内の HSMs の数と状態を確認するには、[AWS CloudHSM コンソール](#)または [DescribeClusters オペレーション](#)を使用します。[DescribeClusters](#) HSM をクラスターに追加するには、AWS CloudHSM コンソールまたは [CreateHsm オペレーション](#)を使用します。
- AWS CloudHSM キーストアに関連付けられた AWS CloudHSM クラスターが削除されました。この問題を解決するには、元のクラスターのバックアップ、または元のクラスターの作成に使用されたバックアップなど、元のクラスターに関連する [バックアップからクラスターを作成](#)します。次に、カスタムキーストアの設定で、[クラスター ID を編集](#)します。手順については、「[KMS キーの削除されたキーマテリアルを復旧するには](#)」を参照してください。
- カスタムキーストアに関連付けられた AWS CloudHSM クラスターには、使用可能な PKCS #11 セッションがありませんでした。通常これは、トラフィックの処理に追加のセッションが必要な大規模なバーストトラフィックが起きている間に発生します。PKCS #11 セッションに関するエラーメッセージで `KMSInternalException` に応答するときは、リクエストをいったん取り消してから、改めて試行します。

接続障害の修復方法

[AWS CloudHSM キーストアをクラスターに接続](#)しようとしてもオペレーションが失敗すると、AWS CloudHSM キーストアの接続状態は `FAILED` に変わります。AWS CloudHSM キーストアの接続状態を確認するには、AWS KMS コンソールまたは [DescribeCustomKeyStores](#) オペレーションを使用します。

また、簡単に検出できるクラスター設定エラーが原因で接続の試行がすぐに失敗することがあります。この場合、接続ステータスは `DISCONNECTED` のままです。これらのエラーは、試行が失敗した理由を説明するエラーメッセージまたは [例外](#) を返します。例外の説明と [クラスターの要件](#)を確認し、問題を修正し、必要に応じて [AWS CloudHSM キーストアを更新](#)して、再度接続を試みます。

接続ステータスが `FAILED` の場合は、[DescribeCustomKeyStores](#) オペレーションを実行してレスポンスの `ConnectionErrorCode` 要素を確認します。

Note

AWS CloudHSM キーストアの接続状態が `FAILED` の場合、再接続を試みる前に [AWS CloudHSM キーストアを切断](#)する必要があります。AWS CloudHSM キーストアを `FAILED` 接続状態に接続することはできません。

- `CLUSTER_NOT_FOUND` は、指定された AWS CloudHSM クラスター ID を持つクラスターが AWS KMS に見つからないことを示します。これは、誤ったクラスター ID が API オペレーションに提供されたか、クラスターが削除されて置き換えられなかったために発生する可能性があります。このエラーを修正するには、AWS CloudHSM コンソールや [DescribeClusters](#) オペレーションなどを使用して、クラスター ID を確認します。クラスターが削除された場合は、元の [バックアップからクラスターを作成します](#)。次に、[AWS CloudHSM キーストアを切断](#)し、[AWS CloudHSM キーストアクラスター ID 設定を編集](#)して、[AWS CloudHSM キーストアをクラスターに再接続](#)します。
- `INSUFFICIENT_CLOUDHSM_HSMs` は、関連付けられた AWS CloudHSM クラスターに HSMs が含まれていないことを示します。クラスターに接続するには、少なくとも 1 つの HSM を持っている必要があります。クラスター内の HSM の数を調べるには、[DescribeClusters](#) オペレーションを使用します。このエラーを解決するには、クラスターに [少なくとも 1 つの HSM を追加します](#)。複数の HSM を追加する場合は、別のアベイラビリティゾーンでそれらを作成することをお勧めします。
- `INSUFFICIENT_FREE_ADDRESSES_IN_SUBNET` は、AWS CloudHSM クラスターに関連付けられた少なくとも 1 つのプライベートサブネットに使用可能な IP アドレスがないため、AWS

CloudHSM キーストアをクラスターに接続 AWS KMS できなかったことを示します。 <https://docs.aws.amazon.com/cloudhsm/latest/userguide/create-subnets.html> AWS CloudHSM キーストア接続には、関連付けられたプライベートサブネットごとに 1 つの空き IP アドレスが必要ですが、2 つが推奨されます。

既存のサブネットに [IP アドレスを追加できません](#) (CIDR ブロック)。可能であれば、未使用の EC2 インスタンスや Elastic Network Interface など、サブネット内の IP アドレスを使用している他のリソースを移動または削除します。それ以外の場合は、[より多くの空きアドレス空間](#)を持つ新規または既存のプライベートサブネットを使用して、クラスターの[最近のバックアップからクラスターを作成できます](#)。AWS CloudHSM 次に、新しいクラスターを AWS CloudHSM キーストアに関連付けるには、[カスタムキーストアを切断](#)し、AWS CloudHSM キーストアの[クラスター ID を新しいクラスターの ID に変更](#)して、再度接続を試みます。

 Tip

[kmsuser パスワードのリセット](#)を回避するには、AWS CloudHSM クラスターの最新のバックアップを使用します。

- INTERNAL_ERROR は、内部エラーのために リクエストを完了 AWS KMS できなかったことを示します。リクエストを再試行します。ConnectCustomKeyStore リクエストの場合は、再度接続を試みる前に AWS CloudHSM キーストアを切断します。
- INVALID_CREDENTIALS は、正しいkmsuserアカウントパスワードがないため、が関連付けられた AWS CloudHSM クラスターにログイン AWS KMS できないことを示します。このエラーのヘルプについては、「[無効な kmsuser 認証情報の修正方法](#)」を参照してください。
- NETWORK_ERRORS は通常、一時的なネットワークの問題を示します。[AWS CloudHSM キーストアを切断](#)し、数分待ってから、もう一度接続を試みます。
- SUBNET_NOT_FOUND は、AWS CloudHSM クラスター設定の少なくとも 1 つのサブネットが削除されたことを示します。がクラスター設定内のすべてのサブネットを検出 AWS KMS できない場合、は AWS CloudHSM キーストアを AWS CloudHSM クラスターに接続しようとして失敗します。

このエラーを修正するには、同じ[クラスターの最近のバックアップからクラスターを作成します](#)。AWS CloudHSM (このプロセスでは、VPC とプライベートサブネットを持つ新しいクラスター設定が作成されます)。新しいクラスターが [カスタムキーストアの要件](#)を満たしていることを確認し、新しいクラスター ID を書き留めます。次に、新しいクラスターを AWS CloudHSM キーストアに関連付けるには、[カスタムキーストアを切断](#)し、AWS CloudHSM キーストアの[クラスター ID を新しいクラスターの ID に変更](#)して、再度接続を試みます。

i Tip

[kmsuser パスワードのリセット](#)を回避するには、AWS CloudHSM クラスターの最新のバックアップを使用します。

- USER_LOCKED_OUT は、失敗したパスワードの試行回数が多すぎるため、[kmsuser Crypto User \(CU\) アカウント](#)が関連付けられた AWS CloudHSM クラスターからロックアウトされたことを示します。このエラーのヘルプについては、「[無効な kmsuser 認証情報の修正方法](#)」を参照してください。

このエラーを修正するには、[AWS CloudHSM キーストアを切断](#)し、CloudHSM CLI の[ユーザー change-password](#) コマンドを使用して kmsuser アカウントのパスワードを変更します。次に、[カスタムキーストアの kmsuser のパスワード設定](#)を編集し、接続し直してみてください。ヘルプについては、「[無効な kmsuser 認証情報の修正方法](#) トピック」で説明されている手順を使用してください。

- USER_LOGGED_IN は、CU kmsuser アカウントが関連付けられた AWS CloudHSM クラスターにログインしていることを示します。これにより、AWS KMS は kmsuser アカウントのパスワードを更新し、クラスターにログインできなくなります。このエラーを修正するには、クラスターから kmsuser CU をログアウトします。クラスターにログインするように kmsuser パスワードを変更した場合は、キーストアのパスワード値も更新する必要があります AWS CloudHSM。ヘルプについては、「[ログアウトして再接続する方法](#)」を参照してください。
- USER_NOT_FOUND は、が関連付けられた AWS CloudHSM クラスターで CU kmsuser アカウントを見つけ AWS KMS られないことを示します。このエラーを修正するには、クラスターに [CU kmsuser アカウントを作成し、キーストアのキーストアパスワード値を更新します](#)。AWS CloudHSM ヘルプについては、「[無効な kmsuser 認証情報の修正方法](#)」を参照してください。

暗号化オペレーションの失敗に対応するには

カスタムキーストアで KMS キーを使用する暗号化オペレーション

は、KMSInvalidStateException で失敗することがあります。次のエラーメッセージには KMSInvalidStateException が関連することがあります。

KMS は CloudHSM クラスターと通信できません。これは一時的なネットワークの問題である可能性があります。このエラーが繰り返し表示される場合は、AWS CloudHSM クラスターの VPC のネットワーク ACLs とセキュリティグループのルールが正しいことを確認します。

- これは HTTPS 400 エラーですが、一時的なネットワークの問題が原因である可能性があります。対応するには、まずリクエストを再試行します。それでも失敗する場合は、ネットワークコンポーネントの設定を調べます。このエラーはほとんどの場合、発信トラフィックをブロックしているファイアウォールルールや VPC セキュリティグループルールなど、ネットワークコンポーネントの誤った設定が原因です。たとえば、KMS は IPv6 経由でクラスターと AWS CloudHSM 通信できません。前提条件の詳細については、「」を参照してください[AWS CloudHSM キーストアを作成する](#)。

kmsuser がロックアウトされているため、KMS は AWS CloudHSM クラスターと通信できません。このエラーが繰り返し表示される場合は、AWS CloudHSM キーストアを切断し、kmsuser アカウントのパスワードをリセットします。カスタムキーストアの kmsuser のパスワードを更新して、リクエストを再試行してください。

- このエラーメッセージは、失敗したパスワードの試行回数が多すぎるため、[kmsuser Crypto User \(CU\) アカウント](#)が関連付けられた AWS CloudHSM クラスターからロックアウトされたことを示します。このエラーのヘルプについては、「[切断してログインする方法](#)」を参照してください。

無効な kmsuser 認証情報の修正方法

[AWS CloudHSM キーストアを接続すると](#)、は [kmsuserCrypto User](#) (CU) として関連する AWS CloudHSM クラスター AWS KMS にログインします。AWS CloudHSM キーストアが切断されるまでログインしたままになります。[DescribeCustomKeyStores](#) 応答は次の例に示すように、FAILED の ConnectionState、および INVALID_CREDENTIALS の ConnectionErrorCode 値を示しています。

AWS CloudHSM キーストアを切断してkmsuserパスワードを変更すると、AWS KMS は CU kmsuser アカウントの認証情報を使用して AWS CloudHSM クラスターにログインできません。その結果、AWS CloudHSM キーストアの接続試行はすべて失敗します。DescribeCustomKeyStores レスポンスは、次の例に示すように、INVALID_CREDENTIALS の FAILED と ConnectionErrorCode の値の ConnectionState を示します。

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleKeyStore
{
  "CustomKeyStores": [
    {
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionErrorCode": "INVALID_CREDENTIALS"
```

```

    "CustomKeyStoreId": "cks-1234567890abcdef0",
    "CustomKeyStoreName": "ExampleKeyStore",
    "TrustAnchorCertificate": "<certificate string appears here>",
    "CreationDate": "1.499288695918E9",
    "ConnectionState": "FAILED"
  ],
}
```

また、誤ったパスワードでクラスターにログインしようとして 5 回失敗すると、AWS CloudHSM はユーザーアカウントをロックします。このクラスターにログインするには、アカウントのパスワードを変更する必要があります。

が kmsuserCU としてクラスターにログインしようとしたときにロックアウトレスポンス AWS KMS を取得すると、AWS CloudHSM キーストアの接続リクエストは失敗します。[DescribeCustomKeyStores](#) 応答には、次の例に示すように、FAILED の ConnectionState、および USER_LOCKED_OUT の ConnectionErrorCode 値が含まれます。

```

$ aws kms describe-custom-key-stores --custom-key-store-name ExampleKeyStore
{
  "CustomKeyStores": [
    {
      "CloudHsmClusterId": "cluster-1a23b4cdefg",
      "ConnectionErrorCode": "USER_LOCKED_OUT",
      "CustomKeyStoreId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleKeyStore",
      "TrustAnchorCertificate": "<certificate string appears here>",
      "CreationDate": "1.499288695918E9",
      "ConnectionState": "FAILED"
    }
  ],
}
```

これらの条件を修復するには、次の手順を実行します。

1. [AWS CloudHSM キーストアを切断](#)します。
2. [DescribeCustomKeyStores](#) オペレーションを実行し、レスポンスの ConnectionErrorCode 要素の値を表示します。
 - ConnectionErrorCode の値が INVALID_CREDENTIALS の場合は、kmsuser アカウントの現在のパスワードを特定します。必要に応じて、CloudHSM CLI の [\[user change-password\]](#) コマンドを使用して、パスワードを既知の値に設定します。

- `ConnectionErrorCode` 値が `USER_LOCKED_OUT` の場合、CloudHSM CLI の [\[user change-password\]](#) コマンドを使用して `kmsuser` パスワードを変更する必要があります。
3. [kmsuser パスワード設定を編集して](#)、クラスター内の現在の `kmsuser` パスワードと一致させます。このアクションは、AWS KMS にクラスターにログインするために使用するパスワードを指示します。クラスターの `kmsuser` パスワードは変更されません。
 4. [カスタムキーストアを接続します](#)。

孤立したキーマテリアルを削除する方法

AWS CloudHSM キーストアから KMS キーの削除をスケジュールした後、関連する AWS CloudHSM クラスターから対応するキーマテリアルを手動で削除する必要がある場合があります。

AWS CloudHSM キーストアで KMS キーを作成する AWS KMS と、は で KMS キーメタデータ AWS KMS を作成し、関連付けられた AWS CloudHSM クラスターでキーマテリアルを生成します。AWS CloudHSM キーストアで KMS キーの削除をスケジュールすると、待機期間の後、は KMS キーメタデータ AWS KMS を削除します。次に、AWS KMS は AWS CloudHSM クラスターから対応するキーマテリアルを削除するために最善を尽くします。AWS CloudHSM キーストアから切断された場合や `kmsuser` パスワードが変更された場合など、がクラスターにアクセス AWS KMS できない場合、試行は失敗する可能性があります。AWS KMS はクラスターバックアップからキーマテリアルを削除しようとしません。

AWS KMS は、AWS CloudTrail ログの `DeleteKey` イベントエントリでクラスターからキーマテリアルを削除しようとした結果を報告します。この結果は、次のエントリ例に示されているように、`additionalEventData` 要素の `backingKeysDeletionStatus` 要素に表示されます。エントリには、KMS キー ARN、AWS CloudHSM クラスター ID、キーマテリアルの ID (`backing-key-id`) も含まれます。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "accountId": "111122223333",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2021-12-10T14:23:51Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "DeleteKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
```

```

    "requestParameters": null,
    "responseElements": {
      "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    "additionalEventData": {
      "customKeyStoreId": "cks-1234567890abcdef0",
      "clusterId": "cluster-1a23b4cdefg",
      "backingKeys": "[{\"backingKeyId\":\"backing-key-id\"}]",
      "backingKeysDeletionStatus": "[{\"backingKeyId\":\"backing-key-id\",
\"deletionStatus\":\"FAILURE\"}]"]
    },
    "eventID": "c21f1f47-f52b-4ffe-bff0-6d994403cf40",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:eu-west-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management"
  }
}

```

メモ

次の手順では、AWS CloudHSM クライアント SDK 5 コマンドラインツール [CloudHSM CLI](#) を使用します。CloudHSM CLI は、key-handle を key-reference に置き換えます。2025 年 1 月 1 日、AWS CloudHSM はクライアント SDK 3 コマンドラインツール、CloudHSM 管理ユーティリティ (CMU)、およびキー管理ユーティリティ (KMU) のサポートを終了します。Client SDK 3 コマンドラインツールと Client SDK 5 コマンドラインツールの違いの詳細については、「AWS CloudHSM ユーザーガイド」の「[Client SDK 3 CMU および KMU から Client SDK 5 CloudHSM CLI への移行](#)」を参照してください。

次の手順は、関連付けられた AWS CloudHSM クラスターから孤立したキーマテリアルを削除する方法を示しています。

1. AWS CloudHSM キーストアがまだ切断されていない場合は、「」の説明に従って[ログイン](#)します[切断してログインする方法](#)。

 Note

カスタムキーストアが切断されている間は、カスタムキーストアで KMS キーを作成したり、暗号化オペレーションで既存の KMS キーを使用したりする試みはすべて失敗します。このオペレーションにより、ユーザーが機密データを保存したりアクセスしたりすることを防ぐことができます。

2. CloudHSM CLI の [key delete](#) コマンドを使用して、クラスター内の HSM から キーを削除します。

AWS CloudHSM キーストアの KMS キーを使用した暗号化オペレーションのすべての CloudTrail ログエントリには、customKeyStoreIdおよびを含む additionalEventDataフィールドが含まれますbackingKey。backingKeyId フィールドで返される値は、CloudHSM キーの id 属性です。CloudTrail ログ内で特定された孤立したキーマテリアルを削除するには、[key delete] オペレーションを id でフィルタリングすることをお勧めします。

AWS CloudHSM はbackingKeyId、値を 16 進値として認識します。id でフィルタリングするには、backingKeyId の先頭に 0x を付加する必要があります。例えば、CloudTrail ログ内の backingKeyId が 1a2b3c45678abcdef の場合、0x1a2b3c45678abcdef でフィルタリングします。

次の例では、クラスター内の HSM から特定のキーを削除します。backing-key-id は CloudTrail ログエントリにリストされています。このコマンドを実行する前に、サンプル backing-key-id をお使いのアカウントにある有効な値に置き換えてください。

```
aws-cloudhsm key delete --filter attr.id="0x<backing-key-id>"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. 「」の説明に従って、ログアウトし、AWS CloudHSM キーストアを再接続します[ログアウトして再接続する方法](#)。

KMS キーの削除されたキーマテリアルを復旧するには

のキーマテリアル AWS KMS key が削除された場合、KMS キーは使用できず、KMS キーで暗号化されたすべての暗号文を復号することはできません。これは、キーストア内の KMS キーの AWS CloudHSM キーマテリアルが関連付けられた AWS CloudHSM クラスターから削除された場合に発生する可能性があります。ただし、キーマテリアルを復元することは可能な場合もあります。

AWS CloudHSM キーストアで AWS KMS key (KMS キー) を作成すると、AWS KMS は関連付けられた AWS CloudHSM クラスターにログインし、KMS キーのキーマテリアルを作成します。また、AWS CloudHSM キーストアが接続されている限り、パスワードのみを知っている値に変更し、ログインしたままになります。キー所有者、つまりキーを作成した CU のみがキーを削除できるため、誤って HSM からキーが削除されることはありません。

ただし、KMS キーのキーマテリアルがクラスター内の HSM から削除されると、KMS キーのキーステータスは最終的に UNAVAILABLE に変わります。暗号化オペレーションに KMS キーを使用しようとすると、KMSInvalidStateException の例外によりオペレーションは失敗します。最も重大なのは、KMS キーで暗号化されたデータが復号できないことです。

特定の状況では、キーマテリアルを含む [バックアップからクラスターを作成することで](#)、削除されたキーマテリアルを回復できます。この方法は、キーが存在していて削除される前に少なくとも 1 つのバックアップが作成された場合にのみ機能します。

キーマテリアルを復旧するには、次の手順を実行します。

1. キーマテリアルが格納されているクラスターバックアップを見つけます。バックアップには、クラスターとその暗号化されたデータをサポートするために必要なすべてのユーザーとキーも含まれている必要があります。

[DescribeBackups](#) オペレーションを使用して、クラスターのバックアップを一覧表示します。バックアップのタイムスタンプを使用すると、バックアップの選択に役立ちます。AWS CloudHSM キーストアに関連付けられているクラスターに出力を制限するには、次の例に示すように、`Filters` パラメータを使用します。

```
$ aws cloudhsmv2 describe-backups --filters clusterIds=<cluster ID>
{
  "Backups": [
    {
      "ClusterId": "cluster-1a23b4cdefg",
      "BackupId": "backup-9g87f6edcba",
      "CreateTimestamp": 1536667238.328,
      "BackupState": "READY"
```



```
    },  
    ...  
  ]  
}
```

2. [選択したバックアップからクラスタを作成します](#)。バックアップに削除されたキーおよびクラスターに必要な他のユーザーとキーが含まれていることを確認します。
3. [AWS CloudHSM キーストアを切断](#)して、そのプロパティを編集できるようにします。
4. AWS CloudHSM キーストアの [クラスター ID を編集](#)します。バックアップから作成したクラスターのクラスター ID を入力します。クラスターは元のクラスターとバックアップ履歴を共有するため、新しいクラスター ID が有効である必要があります。
5. [AWS CloudHSM キーストアを再接続します](#)。

kmsuser としてログインする方法

キーストアの AWS CloudHSM クラスターで AWS CloudHSM キーマテリアルを作成および管理するために、AWS KMS は [kmsuserCrypto User \(CU\) アカウント](#) を使用します。クラスターに [CU kmsuser アカウントを作成し](#)、AWS CloudHSM キーストアの作成 AWS KMS 時に パスワードを指定します。

一般的に、は kmsuser アカウント AWS KMS を管理します。ただし、一部のタスクでは、AWS CloudHSM キーストアを切断し、CU kmsuser としてクラスターにログインし、[CloudHSM コマンドラインインターフェイス \(CLI\)](#) を使用する必要があります。

Note

カスタムキーストアが切断されている間は、カスタムキーストアで KMS キーを作成したり、暗号化オペレーションで既存の KMS キーを使用したりする試みはすべて失敗します。このオペレーションにより、ユーザーが機密データを保存したりアクセスしたりすることを防ぐことができます。

このトピックでは、[AWS CloudHSM キーストアを切断して としてログイン](#)し kmsuser、AWS CloudHSM コマンドラインツールを実行し、[キー AWS CloudHSM ストアをログアウトして再接続する](#)方法について説明します。

トピック

- [切断してログインする方法](#)

- [ログアウトして再接続する方法](#)

切断してログインする方法

関連するクラスターに kmsuser Crypto User (CU) としてログインする必要がある際には、次の手順を実行します。

メモ

次の手順では、AWS CloudHSM クライアント SDK 5 コマンドラインツール [CloudHSM CLI](#) を使用します。CloudHSM CLI は、key-handle を key-reference に置き換えます。2025 年 1 月 1 日、AWS CloudHSM はクライアント SDK 3 コマンドラインツール、CloudHSM 管理ユーティリティ (CMU)、およびキー管理ユーティリティ (KMU) のサポートを終了します。Client SDK 3 コマンドラインツールと Client SDK 5 コマンドラインツールの違いの詳細については、「AWS CloudHSM ユーザーガイド」の「[Client SDK 3 CMU および KMU から Client SDK 5 CloudHSM CLI への移行](#)」を参照してください。

1. まだ切断されていない場合は、AWS CloudHSM キーストアを切断します。AWS KMS コンソールまたは AWS KMS API を使用できます。

AWS CloudHSM キーが接続されている間、AWS KMS は `kmsuser` としてログインします。これにより、`kmsuser` としてログインしたり、`kmsuser` パスワードを変更することができなくなります。

例えば、このコマンドは、[DisconnectCustomKeyStore](#) を使用して、サンプルキーストアを切断します。サンプル AWS CloudHSM キーストア ID を有効なキーストア ID に置き換えます。

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

2. [login] コマンドを使用して、管理者としてログインします。「AWS CloudHSM ユーザーガイド」の「[CloudHSM CLI の使用](#)」セクションに記載されている手順を使用します。

```
aws-cloudhsm > login --username admin --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
```

```
    "role": "admin"
  }
}
```

- CloudHSM CLI の [ユーザー change-password](#) コマンドを使用して、kmsuser アカウントのパスワードを既知のパスワードに変更します (AWS CloudHSM キーストアを接続するとパスワードがAWS KMS ロータションされます)。パスワードは 7~32 の英数字で構成する必要があります。大文字と小文字が区別され、特殊文字を含めることはできません。
- 設定したパスワードを使用して kmsuser としてログインします。詳細な手順については、「AWS CloudHSM ユーザーガイド」の「[CloudHSM CLI の使用](#)」のセクションを参照してください。

```
aws-cloudhsm > login --username kmsuser --role crypto-user
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "kmsuser",
    "role": "crypto-user"
  }
}
```

ログアウトして再接続する方法

kmsuser Crypto User (CU) ユーザーとしてログアウトしてキーストアを再接続する際には、次の手順を使用します。

メモ

次の手順では、AWS CloudHSM クライアント SDK 5 コマンドラインツール [CloudHSM CLI](#) を使用します。CloudHSM CLI は、key-handle を key-reference に置き換えます。2025 年 1 月 1 日、AWS CloudHSM はクライアント SDK 3 コマンドラインツール、CloudHSM 管理ユーティリティ (CMU)、およびキー管理ユーティリティ (KMU) のサポートを終了します。Client SDK 3 コマンドラインツールと Client SDK 5 コマンドラインツールの違いの詳細については、「AWS CloudHSM ユーザーガイド」の「[Client SDK 3 CMU および KMU から Client SDK 5 CloudHSM CLI への移行](#)」を参照してください。

1. タスクを実行後、CloudHSM CLI の [logout](#) コマンドを使用してログアウトします。ログアウトしないと、AWS CloudHSM キーストアの再接続は失敗します。

```
aws-cloudhsm  logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

2. カスタムキーストアの [kmsuser パスワード設定を編集します](#)。

これにより AWS KMS、クラスターkmsuser内の の現在のパスワードが通知されます。このステップを省略すると、 は としてクラスターにログイン AWS KMS できずkmsuser、カスタムキーストアの再接続の試行はすべて失敗します。 [UpdateCustomKeyStore](#) オペレーションの AWS KMS コンソールまたは KeyStorePasswordパラメータを使用できます。

たとえば、このコマンドは、現在のパスワードが AWS KMS であることを に指示しますtempPassword。例のパスワードを実際のパスワードに置き換えます。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --key-store-password tempPassword
```

3. AWS KMS キーストアを AWS CloudHSM クラスターに再接続します。サンプル AWS CloudHSM キーストア ID を有効なキーストア ID に置き換えます。接続プロセス中に、はkmsuserパスワードを自分だけが知っている値 AWS KMS に変更します。

[ConnectCustomKeyStore](#) オペレーションはすばやく返されますが、接続プロセスに長時間かかる場合があります。最初のレスポンスは、接続プロセスの成功を示していません。

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

4. [DescribeCustomKeyStores](#) オペレーションを使用して、AWS CloudHSM キーストアが接続されていることを確認します。サンプル AWS CloudHSM キーストア ID を有効なキーストア ID に置き換えます。

この例では、接続状態フィールドは、AWS CloudHSM キーストアが接続されたことを示しています。

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
{
```

```
"CustomKeyStores": [  
  "CustomKeyId": "cks-1234567890abcdef0",  
  "CustomKeyName": "ExampleKeyStore",  
  "CloudHsmClusterId": "cluster-1a23b4cdefg",  
  "TrustAnchorCertificate": "<certificate string appears here>",  
  "CreationDate": "1.499288695918E9",  
  "ConnectionState": "CONNECTED"  
],  
}
```

外部キーストア

外部キーストアを使用すると、外部で暗号化キーを使用して AWS リソースを保護できます AWS。この高度な機能は、ユーザーが管理する外部のキー管理システムに保存された暗号化キーで保護しなければならない、規制対象のワークロード向けに設計されています。外部キーストアは、お客様が所有および管理する外部キーマテリアルで暗号化する機能など AWS、内のデータを主権的に制御するための [AWS デジタル主権の約束](#) をサポートします AWS。

外部キーストアは、外部で所有および管理している外部キーマネージャーによってバックアップされた [カスタムキーストア](#) です AWS。外部キーマネージャーは、物理的または仮想的なハードウェアセキュリティモジュール (HSM) である場合もあれば、暗号化キーの生成や使用を行うハードウェアベースまたはソフトウェアベースのシステムである場合もあります。外部キーストアの KMS キーを使用した暗号化と復号のオペレーションは、外部キーマネージャーがユーザーの暗号化キーマテリアルを使って実行します。この機能は Hold Your Own Key (HYOK) と呼ばれています。

AWS KMS は外部キーマネージャーと直接やり取りすることではなく、キーを作成、表示、管理、または削除することもできません。代わりに、は、指定した [外部キーストアプロキシ](#) (XKS プロキシ) ソフトウェアとのみ AWS KMS やり取りします。外部キーストアプロキシは、AWS KMS と外部キーマネージャー間のすべての通信を仲介します。からのすべてのリクエスト AWS KMS を外部キーマネージャーに送信し、外部キーマネージャーからのレスポンスを に送信します AWS KMS。外部キーストアプロキシは、からの一般的なリクエスト AWS KMS を、外部キーマネージャーが理解できるベンダー固有の形式に変換し、さまざまなベンダーのキーマネージャーと外部キーストアを使用できるようにします。

外部キーストアの KMS キーは、[AWS Encryption SDK](#) など、クライアント側の暗号化に使用できます。ただし、外部キーストアはサーバー側の暗号化にとって重要なリソースであり、外部の暗号化キー AWS のサービスを使用して複数の の AWS リソースを保護できます AWS。AWS のサービス対称暗号化の [カスタマーマネージドキー](#) をサポートする は、外部キーストアの KMS キーもサポートしています。サービスサポートの詳細については、「[AWS サービス統合](#)」を参照してください。

外部キーストアを使用すると、暗号化キーを外部に保存して使用する必要がある規制対象のワークロード AWS KMS に 使用できます AWS。ただし、こうした使用は標準の責任共有モデルから大きく逸脱するものであり、オペレーション上の負担が増えることになります。可用性とレイテンシーに関するリスクが増えれば、そのリスクは、大半のお客様にとって外部キーストアのセキュリティにおける利点を上回るものとなります。

外部キーストアを使用すると、ユーザーは、信頼の基点を制御することができます。外部キーストアの KMS キーで暗号化されたデータは、ユーザーが管理する外部キーマネージャーでしか復号することはできません。外部キーストアを切断したり、外部キーストアプロキシから外部キーマネージャーを切断したりするなど、外部キーマネージャーへのアクセスを一時的に取り消すと、は、復元するまで暗号化キーへのすべてのアクセス AWS を失います。その間、ユーザーの KMS キーで暗号化された暗号文は復号できません。外部キーマネージャーへのアクセスを永久に無効にすると、外部キーストアの KMS キーで暗号化された暗号文は、すべて回復不能になります。唯一の例外は、KMS [キーで保護されたデータ](#) キーを短時間キャッシュする AWS サービスです。これらのデータキーは、リソースを非アクティブにするまで、またはキャッシュの有効期限が切れるまで、機能し続けます。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

外部キーストアは、暗号化キーがユーザーのみの管理下にあり、アクセスできない必要がある、規制対象のワークロードのいくつかのユースケースのブロックを解除します AWS。しかし、こうした使用はクラウドベースのインフラストラクチャの運用方法を大きく変えるもので、責任共有モデルに著しい変化をもたらします。ほとんどのワークロードでオペレーションの負担が増えることになり、可用性とパフォーマンスのリスクが増え、そのリスクは、外部キーストアを利用するセキュリティ上の利点を上回ります。

自分は外部キーストアを使う必要がありますか？

ほとんどのユーザーにとって、[FIPS 140-3 セキュリティレベル 3 検証済みハードウェアセキュリティモジュール](#)で保護されているデフォルトの AWS KMS キーストアは、セキュリティ、コントロール、および規制要件を満たしています。外部キーストアを使用するユーザーには、多額のコスト、メンテナンスやトラブルシューティングなどの負担、レイテンシー、可用性、信頼性のリスクが生じます。

外部キーストアの使用を検討する際は、代わりの手段についても理解しておきましょう。例えば、ユーザーが所有し管理する、AWS CloudHSM クラスターでバックアップされた [AWS CloudHSM キーストア](#)や、ユーザーが自身の HSM で生成し、必要に応じて KMS キーから削除できる、[インポートされたキーマテリアル](#)を使う KMS キーなどです。特に、有効期限が非常に短いキーマテリアル

ルをインポートすると、パフォーマンスや可用性のリスクなしに、同様のレベルの制御が可能になる場合があります。

以下の要件がある場合は、外部キーストアが組織にとって適切なソリューションとなる可能性があります。

- オンプレミスのキーマネージャーまたは AWS 自分が管理する 以外のキーマネージャーで暗号化キーを使用する必要があります。
- 暗号化キーが、クラウド以外の場所で、ユーザーが単独で管理して保持されていることを証明する必要があります。
- 暗号化と復号で、独立した認可を受けた暗号化キーを使用する必要があります。
- キーマテリアルは、補助的な、独立した監査パスの対象とする必要があります。

外部キーストアを選択する場合は、その使用を、AWS以外の暗号化キーによる保護を必要とするワークロードに限定します。

責任共有モデル

標準 KMS キーは、AWS KMS を所有および管理する HSMs で生成および使用されるキーマテリアルを使用します。KMS キーのアクセスコントロールポリシーを確立し、KMS キー AWS のサービスを使用してリソースを保護する を設定します。AWS KMS は、KMS キー内のキーマテリアルのセキュリティ、可用性、レイテンシー、耐久性について責任を負います。

外部キーストアの KMS キーは、外部キーマネージャーにおけるキーマテリアルとオペレーションに依存します。そのため、責任のバランスはユーザー側にシフトします。お客様は、外部キーマネージャーの暗号化キーのセキュリティ、信頼性、耐久性、パフォーマンスについて責任を負います。AWS KMS は、リクエストに迅速に対応し、外部キーストアプロキシと通信し、当社のセキュリティ基準を維持する責任があります。すべての外部キーストア暗号文が、少なくとも標準 AWS KMS 暗号文と同じ強度になるように、は AWS KMS まずすべてのプレーンテキストを KMS キーに固有の AWS KMS キーマテリアルで暗号化してから、それを外部キーマネージャーに送信して外部キーで暗号化します。これは、[二重暗号化](#)と呼ばれる手順です。それにより、AWS KMS も外部キーマテリアルの所有者も、二重に暗号化された暗号文を単独で復号することはできなくなります。

ユーザーは、規制およびパフォーマンス基準を満たす外部キーマネージャーの管理、[AWS KMS 外部キーストアプロキシ API 仕様](#)に準拠する外部キーストアプロキシの提供と維持、およびキーマテリアルの可用性と耐久性の確保に責任を負います。また、は外部キーストアの作成、設定、維持もユー

ザーが行います。メンテナンスするコンポーネントが原因でエラーが発生した場合は、AWS サービスが過度に中断することなくリソースにアクセスできるように、エラーを特定して解決する準備を整える必要があります。は、問題の原因と最も可能性の高い解決策を判断するのに役立つ[トラブルシューティングガイド](#) AWS KMS を提供します。

が AWS KMS 外部キーストアに記録する [Amazon CloudWatch メトリクスとディメンション](#)を確認します。では、CloudWatch アラームを作成して外部キーストアをモニタリングし、パフォーマンスと運用の問題の初期兆候を事前に検出できるようにすることを AWS KMS 強くお勧めします。

違いは何か

外部キーストアは、対称暗号化 KMS キーのみをサポートしています。内部では AWS KMS、[アクセスコントロールポリシーの設定](#)やキーの[使用のモニタリング](#)など、他の[カスターマネージドキー](#)を管理するのとはほぼ同じ方法で、外部キーストアで KMS キーを使用および管理します。ユーザーが KMS キーに使用している、外部キーストアの KMS キーを使った暗号化オペレーションをリクエストするときは、同じ API を同じパラメータで使します。料金も、標準の KMS キーと同じです。詳細については、「[外部キーストアの KMS キー](#)」および「[AWS Key Management Service の料金](#)」を参照してください。

ただし、外部キーストアには以下のような原則の変更があります。

- キーオペレーションの可用性、耐久性、レイテンシーの責任はユーザーが負う。
- 外部キーマネージャシステムの開発、購入、オペレーション、ライセンス供与のコストは、ユーザーが負担する。
- から外部キーストアプロキシ AWS KMS へのすべてのリクエストに対して、[独立した認可](#)を実装できます。
- 外部キーストアプロキシのすべてのオペレーションと、AWS KMS リクエストに関連する外部キーマネージャのすべてのオペレーションをモニタリング、監査、ログ記録できます。

開始方法

外部キーストアを作成して管理するには、[外部キーストアプロキシ接続オプションを選択](#)し、[前提条件を構成](#)し、[外部キーストアを作成、設定](#)する必要があります。

クォータ

AWS KMS では、接続状態に関係なく、[キーストアと外部キーストアの両方を含む、各 およびリージョンに最大 10 個のカスタムキーストア](#)を使用できます。AWS アカウント [AWS CloudHSM](#)ま

た、[外部キーストアで KMS キーを使用する](#)ときは、AWS KMS リクエストクォータが適用されます。

外部キーストアプロキシに [VPC プロキシ接続](#)を選択すると、VPC、サブネット、Network Load Balancer などの必要なコンポーネントにもクォータが適用される場合があります。これらのクォータの詳細については、[Service Quotas コンソール](#)を参照してください。

リージョン

ネットワークのレイテンシーを最小限に抑えるには、[外部キーマネージャー](#)に最も近い AWS リージョン に外部キーストアコンポーネントを作成します。可能な場合は、ネットワークラウンドトリップ時間 (RTT) が 35 ミリ秒以下のリージョンを選択します。

外部キーストアは、中国 (北京) と中国 (寧夏) を除き、AWS KMS がサポートされているすべての AWS リージョン でサポートされています。

サポートされていない機能

AWS KMS は、カスタムキーストアで以下の機能をサポートしていません。

- [非対称 KMS キー](#)
- [HMAC KMS キー](#)
- [インポートされたキーマテリアルを持つ KMS キー](#)
- [自動キーローテーション](#)
- [マルチリージョンキー](#)

詳細はこちら:

- AWS ニュースブログ「[AWS KMS 外部キーストアの発表](#)」。

外部キーストアのコンセプト

外部キーストアで使用される基本的な用語とコンセプトについて説明します。

外部キーストア

外部キーストアは、所有および管理 AWS している 外の外部キーマネージャーによってバックアップされた AWS KMS [カスタムキーストア](#)です。外部キーストアの各 KMS キーは、外部キーマネー

ジャーの[外部キー](#)に関連付けられています。外部キーストアの KMS キーを使って暗号化または複合化を行う場合、オペレーションは、外部キーマネージャーでユーザーの外部キーを使用して実行されます。この方法は Hold your Own Keys (HYOK) と呼ばれます。この機能は、暗号化キーを自社の外部キーマネージャーで管理する必要のある組織向けに設計されています。

外部キーストアは、AWS リソースを保護する暗号化キーとオペレーションが、ユーザーが管理する外部キーマネージャーに保持されるようにします。は、データを暗号化および復号化するためのリクエストを外部キーマネージャー AWS KMS に送信しますが、外部キーを作成、削除、管理 AWS KMS することはできません。から外部キーマネージャー AWS KMS へのすべてのリクエストは、ユーザーが提供、所有、管理する[外部キーストアプロキシ](#)ソフトウェアコンポーネントによって仲介されます。

AWS AWS KMS [カスターマネージドキー](#)をサポートする のサービスは、外部キーストアの KMS キーを使用してデータを保護できます。それによりデータは、最終的にユーザーのキーによって、外部キーマネージャーで暗号化オペレーションを使用して保護されます。

外部キーストアの KMS キーは、信頼モデル、[責任共有の取り決め](#)、期待されるパフォーマンスが標準の KMS キーとは根本的に異なります。外部キーストアを使用した場合、ユーザーは、キーマテリアルのセキュリティと整合性、および暗号化のオペレーションに責任を負います。外部キーストアにある KMS キーの可用性とレイテンシーは、ハードウェア、ソフトウェア、ネットワークコンポーネント、そして AWS KMS と外部キーマネージャーとの距離の影響を受けます。また、外部キーマネージャーと、外部キーマネージャーが通信するために必要なネットワークおよびロードバランシングインフラストラクチャに追加のコストが発生する可能性があります。AWS KMS

外部キーストアは、より広範なデータ保護戦略の一環として使用することができます。保護する AWS リソースごとに、外部キーストアに KMS キーを必要とするものと、標準 KMS キーで保護できるものを決定できます。これにより、KMS キーを、特定のデータ分類、アプリケーション、プロジェクト向けに柔軟に選択できます。

外部キーマネージャー

外部キーマネージャーは、256 ビット AES 対称キーを生成し、対称暗号化と復号を実行できる AWS の外部コンポーネントです。外部キーストアの外部キーマネージャーには、物理ハードウェアセキュリティモジュール (HSM)、仮想 HSM、HSM コンポーネントの有無にかかわらずソフトウェアキーマネージャー、のいずれかを使用できます。オンプレミス AWS、ローカルデータセンター、リモートデータセンター、クラウドなど、 の外部に配置できます。外部キーストアは、単一の外部キーマネージャー、または暗号化キーを共有している、複数の関連するキーマネージャーインスタンス (HSM クラスターなど) によってバックアップできます。外部キーストアは、さまざまなベンダーの、さまざまな外部マネージャーをサポートするように設計されています。外部キーマネー

ジャーへの接続の詳細については、「[外部キーストアプロキシとの接続オプションを選択する](#)」を参照してください。

外部キー

外部キーストアの各 KMS キーは、外部キーとも呼ばれる、[外部キーマネージャー](#)にある暗号化キーに関連付けられています。外部キーストアの KMS キーを使って暗号化または複合化を行うとき、暗号化オペレーションは、[外部キーマネージャー](#)によって、ユーザーの外部キーを使用して実行されます。

Warning

外部キーは、KMS キーのオペレーションに不可欠なものです。外部キーを紛失したり削除したりすると、関連付けられた KMS キーで暗号化された暗号文は回復不能になります。

外部キーストアの場合、外部キーは、有効化され、暗号化と復号を実行できる、256 ビット AES キーでなければなりません。外部キー要件の詳細については、「[外部キーストアの KMS キーの要件](#)」を参照してください。

AWS KMS は外部キーを作成、削除、管理することはできません。ユーザーの暗号化キーマテリアルは、外部キーマネージャーの外に出ることはありません。外部キーストアで KMS キーを作成するときは、ユーザーが外部キーの ID (XksKeyId) を指定します。KMS キーに関連付けられた外部キー ID は変更できません。ただし、外部キーマネージャは、外部キー ID に関連付けられたキーマテリアルをローテーションできます。

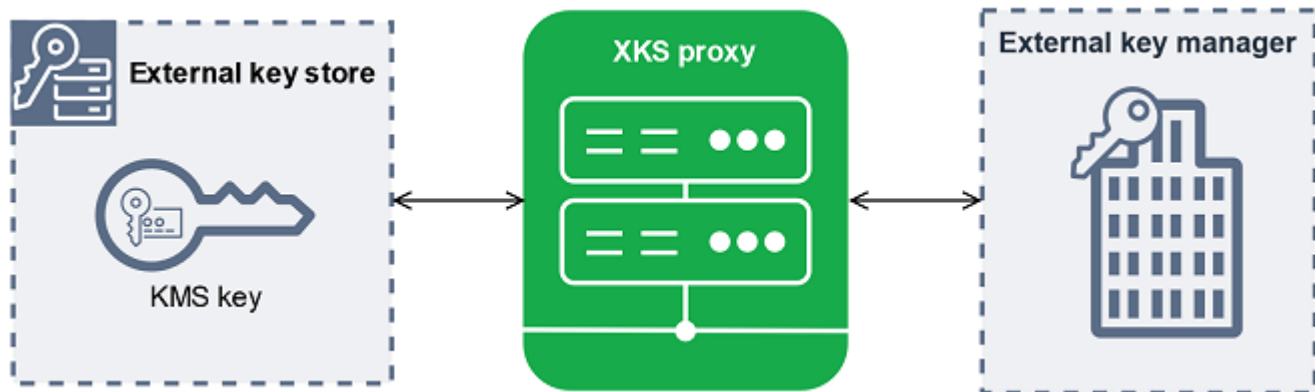
外部キーのほかに、外部キーストアの KMS キーには AWS KMS キーマテリアルもあります。KMS キーで保護されたデータは、最初に AWS KMS キーマテリアル AWS KMS を使用して暗号化され、次に外部キーマネージャーによって外部キーを使用して暗号化されます。この[二重暗号化](#)プロセスにより、KMS キーで保護された暗号文は、AWS KMSのみで保護された暗号文と同等以上の安全性を維持できます。

暗号化キーの多くは、異なる種類の識別子を持ちます。外部キーストアに KMS キーを作成するときは、[外部キーストアプロキシ](#)が外部キーを参照するときに使用する、外部キーの ID を指定します。誤った識別子を使用すると、外部キーで KMS キーを作成するときに失敗します。

外部キーストアプロキシ

外部キーストアプロキシ (「XKS プロキシ」) は、と外部キーマネージャーとの間のすべての通信を仲介する、カスタマー所有 AWS KMS およびカスタマー管理のソフトウェアアプリケーション

です。また、一般的な AWS KMS リクエストをベンダー固有の外部キーマネージャーが理解できる形式に変換します。外部キーストアには、外部キーストアプロキシが必要です。各外部キーストアは、1 つの外部キーストアプロキシに関連付けられています。



AWS KMS は外部キーを作成、削除、管理することはできません。ユーザーの暗号化キーマテリアルは、ユーザーの外部キーマネージャーの外に出ることはありません。AWS KMS と外部キーマネージャー間のすべての通信は、外部キーストアプロキシによって仲介されます。は、外部キーストアプロキシにリクエスト AWS KMS を送信し、外部キーストアプロキシからレスポンスを受け取ります。外部キーストアプロキシは、 から外部キーマネージャー AWS KMS にリクエストを送信し、外部キーマネージャーから にレスポンスを送信する責任があります。AWS KMS

外部キーストアの外部キーストアプロキシを所有および管理して、そのメンテナンスとオペレーションを行う責任は、ユーザーが負います。外部キーストアプロキシは、ベンダーからプロキシアプリケーションを公開または購入するオープンソースの[外部キーストアプロキシ API 仕様](#)に基づいて開発できます。AWS KMS 外部キーストアプロキシは、外部キーマネージャーに含まれている場合があります。プロキシ開発をサポートするために、には、外部キーストアプロキシのサンプル ([aws-kms-xks-proxy](#)) と、外部キーストアプロキシが仕様に準拠していることを検証するテストクライアント ([xks-kms-xksproxy-test-client](#)) AWS KMS も用意されています。

認証先として AWS KMS、プロキシはサーバー側の TLS 証明書を使用します。プロキシに対して認証するために、は SigV4 プロキシ[認証情報](#)を使用して外部キーストアプロキシへのすべてのリクエスト AWS KMS に署名します。オプションで、プロキシは相互 TLS (mTLS) を有効にして、リクエストのみを受け入れることをさらに保証できます AWS KMS。

外部キーストアプロキシは、以下の暗号スイートのうち少なくとも 1 つを含む HTTP/1.1 以降と TLS 1.2 以降をサポートしている必要があります。

- TLS_AES_256_GCM_SHA384 (TLS 1.3)
- TLS_CHACHA20_POLY1305_SHA256 (TLS 1.3)

Note

AWS GovCloud (US) Region は TLS_CHACHA20_POLY1305_SHA256 をサポートしていません。

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (TLS 1.2)
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (TLS 1.2)

外部キーストアの KMS キーを作成し使用するときは、最初に[外部キーストアを外部キーストアプロキシに接続](#)しておく必要があります。外部キーストアは、必要に応じてプロキシから切断することも可能です。切断すると、外部キーストアのすべての KMS キーが[使用できなくなり](#)、いずれの暗号化オペレーションでもそれらを使用できなくなります。

外部キーストアプロキシ接続

外部キーストアプロキシ接続 (「XKS プロキシ接続」) は、が外部キーストアプロキシとの通信 AWS KMS に使用する方法を記述します。

外部キーストアを作成するときにプロキシ接続のオプションを指定すると、そのプロキシ接続は外部キーストアのプロパティになります。プロキシ接続オプションは、カスタムキーストアプロパティを更新することで変更できますが、外部キーストアプロキシが引き続き同じ外部キーにアクセスできることを、確認しておく必要があります。

AWS KMS は、次の接続オプションをサポートしています。

- [パブリックエンドポイント接続](#) — インターネット経由で外部キーストアプロキシのリクエストを、ユーザーが管理するパブリックエンドポイント AWS KMS に送信します。このオプションは簡単に作成し管理することができますが、すべてのインストールのセキュリティ要件を満たしているとは限りません。
- [VPC エンドポイントサービス接続](#) — 作成して維持する Amazon Virtual Private Cloud (Amazon VPC) エンドポイントサービスにリクエスト AWS KMS を送信します。Amazon VPC 内で外部キーストアプロキシをホストするか、の外部で外部キーストアプロキシをホスト AWS し、Amazon VPC を通信にのみ使用できます。

外部キーストアプロキシ接続のオプションの詳細については、「[外部キーストアプロキシとの接続オプションを選択する](#)」を参照してください。

外部キーストアのプロキシ認証の認証情報

外部キーストアプロキシに認証するために、は [署名 V4 \(SigV4\)](#) 認証情報を使用して外部キーストアプロキシへのすべてのリクエスト AWS KMS に署名します。プロキシで認証情報を確立して維持し、外部ストアを作成する AWS KMS ときにこの認証情報を に提供します。

Note

AWS KMS が XKS プロキシへのリクエストに署名するために使用する SigV4 認証情報は、の AWS Identity and Access Management プリンシパルに関連付けられた SigV4 認証情報とは無関係です AWS アカウント。IAM SigV4 認証情報を外部キーストアプロキシに再利用しないでください。

各プロキシ認証の認証情報は、2 つの要素から成ります。外部キーストアを作成するとき、または外部キーストア用にプロキシ認証の認証情報を更新するときは、これら両方の要素を指定する必要があります。

- アクセスキー ID: シークレットアクセスキーを識別します。この ID はプレーンテキストで入力できます。
- シークレットアクセスキー: 認証情報のシークレット部分。 は、認証情報を保存する前に、認証情報内のシークレットアクセスキーを AWS KMS 暗号化します。

[認証情報の設定](#) は、間違った値を入力したとき、プロキシの認証情報を変更するとき、プロキシが認証情報をローテーションするときなどにいつでも編集できます。外部キーストアプロキシへの認証に関する AWS KMS 技術的な詳細については、AWS KMS 「外部キーストアプロキシ API 仕様」の [「認証」](#) を参照してください。

外部キーストアで KMS キーを使用する AWS のサービス を中断することなく認証情報を更新できるようにするには、外部キーストアプロキシが少なくとも 2 つの有効な認証情報をサポートすることをお勧めします AWS KMS。これにより、新しい認証情報を AWS KMS に提供している間も、以前の認証情報が引き続き機能します。

プロキシ認証情報の経過時間を追跡しやすくするために、は Amazon CloudWatch メトリクス [XksProxyCredentialAge](#) AWS KMS を定義します。このメトリクスを使用すれば、認証情報が、設定した有効期限に達したときに通知する、CloudWatch アラームを作成できます。

外部キーストアプロキシが AWS KMS のみに応答することをさらに確実にするため、一部の外部キープロキシは Mutual Transport Layer Security (mTLS) をサポートしています。詳細については、「[mTLS 認証 \(オプション\)](#)」を参照してください。

プロキシ API

AWS KMS 外部キーストアをサポートするには、[外部キーストアプロキシ](#)は、外部キーストアプロキシ APIs 仕様で説明されているように、必要なプロキシ API を実装する必要があります。[AWS KMS](#)これらのプロキシ API リクエストは、がプロキシ AWS KMS に送信する唯一のリクエストです。ユーザーはこれらのリクエストを直接送信することはありませんが、これらについて知っておくと、外部キーストアやそのプロキシで発生する問題を修正する際に役に立つ場合があります。たとえば、は、外部キーストアの [Amazon CloudWatch メトリクス](#)に、これらの API コールのレイテンシーと成功率に関する情報 AWS KMS を含めます。詳細については、「[外部キーストアをモニタリングする](#)」を参照してください。

以下の表には各プロキシ API が一覧表示され、それぞれが説明されています。また、プロキシ API への呼び出しをトリガーする AWS KMS オペレーションと、プロキシ API に関連する AWS KMS オペレーション例外も含まれます。

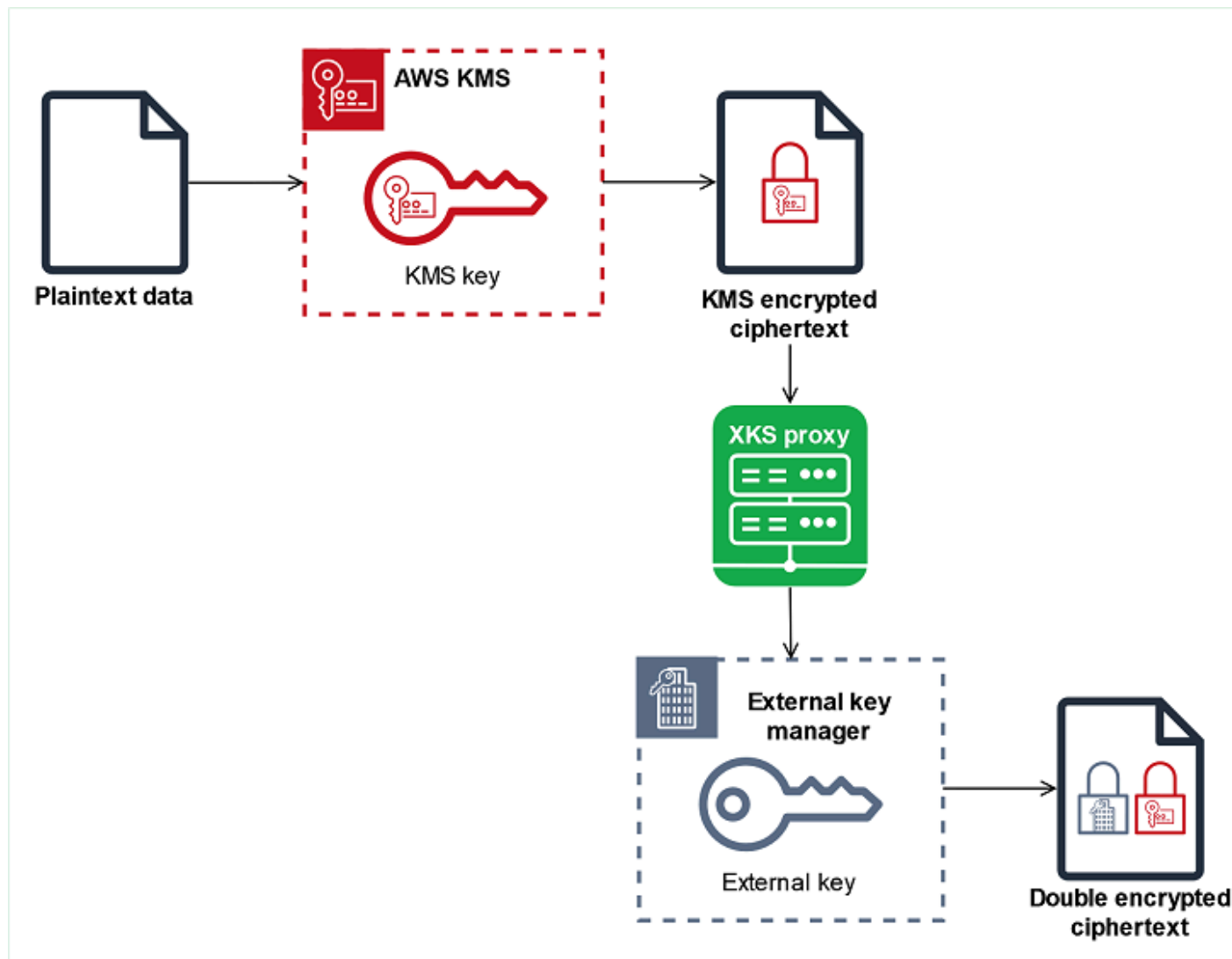
プロキシ API	説明	関連する AWS KMS オペレーション
Decrypt	AWS KMS は復号する暗号文と、使用する 外部キー の ID を送信します。必要な暗号化アルゴリズムは AES_GCM です。	Decrypt 、 ReEncrypt
暗号化	AWS KMS は、暗号化するデータと、使用する 外部キー の ID を送信します。必要な暗号化アルゴリズムは AES_GCM です。	Encrypt 、 GenerateDataKey 、 GenerateDataKeyWithoutPlaintext 、 ReEncrypt
GetHealthStatus	AWS KMS は、プロキシと外部キーマネージャーのステータスに関する情報をリクエストします。 各外部キーマネージャーのステータスは、以下のいずれかになります。	CreateCustomKeyStore (パブリックエンドポイント接続用)、 ConnectCustomKeyStore (VPC エンドポイントサービス接続用) すべての外部キーマネージャーインスタンスが Unavailable である場合、キーストアを作成または

プロキシ API	説明	関連する AWS KMS オペレーション
	• Active: 正常。トラフィックを処理できる • Degraded: 異常。ただし、トラフィックを処理できる • Unavailable : 異常。トラフィックを処理できない	は接続しようとする XksProxyUriUnreachableException で失敗します。
GetKeyMetadata	AWS KMS は、外部キーストアの KMS キーに関連付けられた外部キーに関する情報をリクエストします。 応答には、キーのスペック (AES_256)、キーの使用 ([ENCRYPT, DECRYPT])、外部キーが ENABLED または DISABLED かどうか、が含まれます。	CreateKey キーのスペックが AES_256 でない場合、キーの使用が [ENCRYPT, DECRYPT] でない場合、ステータスが DISABLED である場合、CreateKey オペレーションは XksKeyInvalidConfigurationException により失敗します。

二重暗号化

外部キーストアの KMS キーで暗号化されるデータは、2 回暗号化されます。まず、は KMS AWS KMS キーに固有のキーマテリアルでデータを AWS KMS 暗号化します。次に、AWS KMSで暗号化された暗号文が、[外部キーマネージャー](#)によって[外部キー](#)を使用して暗号化されます。このプロセスは二重暗号化と呼ばれます。

二重暗号化を使用すれば、外部キーストアの KMS キーで暗号化されたデータを、標準の KMS キーで暗号化された暗号文と同等以上の安全性で維持できます。また、 から外部キーストアプロキシ AWS KMS への転送中のプレーンテキストも保護します。二重暗号化を使うことで、暗号文を完全に制御できます。外部キーへの AWS アクセスを外部プロキシ経由で永久に無効にすると、AWSに残っている暗号文は暗号によって実質的に細断されます。



二重暗号化を有効にするには、外部キーストア内の各 KMS キーに 2 つの暗号化バックアップキーを作成します。

- KMS AWS KMS キーに固有のキーマテリアル。このキーマテリアルは、AWS KMS [FIPS 140-3 セキュリティレベル 3](#) 認定ハードウェアセキュリティモジュール (HSMs) でのみ生成され、使用されます。
- 外部キーマネージャーの[外部キー](#)。

二重暗号化には次の効果があります。

- AWS KMS は、外部キーストアプロキシを介して外部キーにアクセスせずに、外部キーストアの KMS キーによって暗号化された暗号文を復号することはできません。

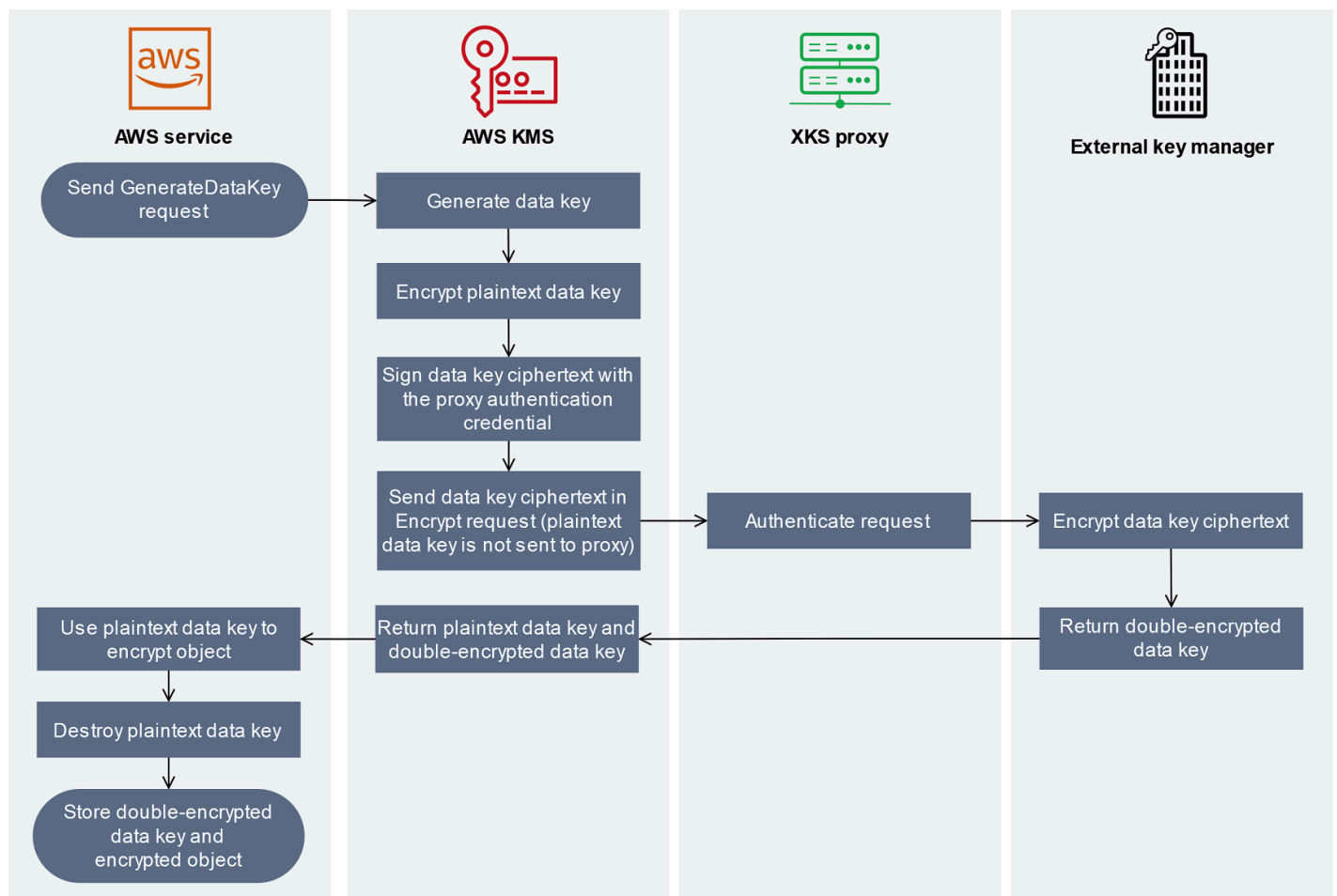
- 外部キーマテリアルがある場合でも AWS、外部の外部キーストアで KMS キーによって暗号化された暗号文を復号することはできません。
- 外部キーストアから削除された KMS キーは、その外部キーマテリアルがある場合でも、再作成できない。各 KMS キーには、対称暗号文に含まれる固有のメタデータがある。新しい KMS キーでは、同じ外部キーマテリアルを使用する場合でも、元のキーで暗号化された暗号文を復号することはできない。

実際の二重暗号化の例については、「[外部キーストアの仕組み](#)」を参照してください。

外部キーストアの仕組み

[外部キーストア](#)、[外部キーストアプロキシ](#)、[外部キーマネージャー](#)は、連携して AWS リソースを保護します。次の手順は、KMS キーで保護された一意のデータキーに基づいて各オブジェクトを暗号化する、一般的な AWS のサービスの暗号化ワークフローを示したものです。この場合、オブジェクトを保護するために、外部キーストアの KMS キーが選択されています。この例では、[二重暗号化](#) AWS KMS を使用して転送中のデータキーを保護し、外部キーストアの KMS キーによって生成された暗号文が、キーマテリアルを含む標準の対称 KMS キーによって暗号化された暗号文と少なくとも同じ強度であることを保証します AWS KMS。

と統合 AWS のサービス する実際の ごとに使用される暗号化方法は AWS KMS 異なります。詳細については、AWS のサービス ドキュメントの「セキュリティ」の項にある「データ保護」のトピックを参照してください。



1. AWS のサービス リソースに新しいオブジェクトを追加します。オブジェクトを暗号化するために、は外部キーストアの KMS キー AWS KMS を使用して [GenerateDataKey](#) リクエストを AWS のサービス に送信します。
2. AWS KMS は 256 ビット対称 [データキー](#) を生成し、外部キーストアプロキシを介してプレーンテキストデータキーのコピーを外部キーマネージャーに送信する準備をします。は、外部キーストアの KMS キーに関連付けられた [AWS KMS キーマテリアル](#) を使用してプレーンテキストデータキーを暗号化することで、[二重暗号化](#) プロセス AWS KMS を開始します。
3. AWS KMS は、外部キーストアに関連付けられた外部キーストアプロキシに [暗号化](#) リクエストを送信します。リクエストには、暗号化されるデータキー暗号文と、KMS [キーに関連付けられている外部](#) キーの ID が含まれます。は、外部キーストアプロキシの [プロキシ認証情報](#) を使用してリクエスト AWS KMS に署名します。

データキーのプレーンテキストのコピーは、外部キーストアプロキシには送信されません。

4. 外部キーストアプロキシがリクエストを認証し、暗号化リクエストを外部のキーマネージャーに渡します。

外部キーストアプロキシの中には、特定の条件下において選択されたプリンシパルのみにオペレーションを許可する、オプションの[認可ポリシー](#)を実装しているものもあります。

5. 外部キーマネージャーは、指定された外部キーを使用して、データキーの暗号文を暗号化します。外部キーマネージャは二重に暗号化されたデータキーを外部キーストアプロキシに返し、外部キーストアプロキシはそれを AWS KMS に返します。
6. AWS KMS は、プレーンテキストのデータキーと、そのデータキーの二重暗号化されたコピーを返します AWS のサービス。
7. AWS のサービス は、プレーンテキストのデータキーを使用してリソースオブジェクトを暗号化し、プレーンテキストのデータキーを破棄して、暗号化されたデータキーを暗号化されたオブジェクトに保存します。

一部の AWS のサービス では、プレーンテキストのデータキーをキャッシュして複数のオブジェクトに使用したり、リソースの使用中に再利用したりできます。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

暗号化されたオブジェクトを復号するには、は暗号化されたデータキーを [Decrypt](#) リクエスト AWS KMS で に送信し直す AWS のサービス 必要があります。暗号化されたデータキーを復号するには、暗号化されたデータキーを外部キーの ID で外部キーストアプロキシ AWS KMS に送信する必要があります。何らかの理由で外部キーストアプロキシへの復号リクエストが失敗した場合、AWS KMS は暗号化されたデータキーを復号できず、は暗号化されたオブジェクトを復号 AWS のサービス できません。

外部キーストアへのアクセスの制御

標準 KMS [キー](#)で使用するすべての AWS KMS アクセスコントロール機能、つまりキーポリシー、[IAM ポリシー](#)、[グラント](#)は、外部キーストアの KMS キーでも同じように機能します。IAM ポリシーを使うことで、外部キーストアを作成し管理するための API オペレーションへのアクセスを、制御できます。IAM ポリシーとキーポリシーを使用して、外部キーストア AWS KMS keys のへのアクセスを制御します。AWS 組織および VPC エンドポイント[ポリシーのサービスコントロール](#)ポリシーを使用して、外部キーストアの KMS キーへのアクセスを制御することもできます。
<https://docs.aws.amazon.com/vpc/latest/privatelink/interface-endpoints.html#edit-vpc-endpoint-policy>

ユーザーとロールには、それらが実行する可能性の高いタスクに必要なアクセス許可のみ、付与することが推奨されます。

トピック

- [外部キーストアマネージャーの承認](#)
- [外部キーストアにおける KMS キーのユーザー認証](#)
- [が外部キーストアプロキシと通信 AWS KMS することを許可する](#)
- [外部キーストアプロキシ認可 \(オプション\)](#)
- [mTLS 認証 \(オプション\)](#)

外部キーストアマネージャーの承認

外部キーストアを作成し管理するプリンシパルには、カスタムキーストアオペレーションへのアクセス許可が必要になります。次のリストは、外部キーストアマネージャーに必要な最小限のアクセス許可です。カスタムキーストアは AWS リソースではないため、他の のプリンシパルに外部キーストアにアクセス許可を付与することはできません AWS アカウント。

- kms:CreateCustomKeyStore
- kms:DescribeCustomKeyStores
- kms:ConnectCustomKeyStore
- kms:DisconnectCustomKeyStore
- kms:UpdateCustomKeyStore
- kms>DeleteCustomKeyStore

外部キーストアを作成するプリンシパルは、外部キーストアコンポーネントを作成し構成するためのアクセス許可が必要になります。プリンシパルは、外部キーストアを自分のアカウントのみで作成できます。[VPC エンドポイントサービスに接続できる](#) 外部キーストアを作成するには、プリンシパルは、次のコンポーネントを作成するためのアクセス許可を持っている必要があります。

- An Amazon VPC
- パブリックサブネットおよびプライベートサブネット
- Network Load Balancer とターゲットグループ
- Amazon VPC エンドポイントサービス

詳細については、「[Amazon VPC の Identity and Access Management](#)」、「[VPC エンドポイントおよび VPC エンドポイントサービスの Identity and Access Management](#)」、「[Elastic Load Balancing API のアクセス許可](#)」を参照してください。

外部キーストアにおける KMS キーのユーザー認証

外部キーストア AWS KMS keys で を作成および管理するプリンシパルには、 で KMS キーを作成および管理するプリンシパルと同じ[アクセス許可](#)が必要です AWS KMS。外部キーストアの KMS キーの、[デフォルトのキーポリシー](#)は、AWS KMSの KMS キーの、デフォルトのキーポリシーと同一です。タグとエイリアスを使用して KMS キーへのアクセスを制御する[属性ベースのアクセス制御](#) (ABAC) は、カスタムキーストアの KMS キーでも同様に有効です。

カスタムキーストアで暗号化オペレーションに KMS キーを使用するプリンシパルには、KMS キーで[暗号化オペレーション](#) ([KMS: Decrypt](#)など) を実行するアクセス許可が必要です。 これらのアクセス権限は、IAM またはキーポリシーで指定できます。ただし、カスタムキーストアで KMS キーを使用するための追加のアクセス許可は必要ありません。

外部キーストアの KMS キーにのみ適用されるアクセス許可を設定するには、値が `EXTERNAL_KEY_STORE` の [kms:KeyOrigin](#) ポリシー条件を使用します。この条件を使用すれば、[kms: CreateKey](#) アクセス許可、または KMS キーリソースに固有のアクセス許可を制限できます。例えば、次の IAM ポリシーを使えば、アタッチされた ID は、KMS が外部キーストアにある限り、アカウントのすべての KMS キーで、指定されたオペレーションを呼び出すことができます。アクセス許可は、外部キーストアの KMS キーと の KMS キーに制限できますが AWS アカウント、アカウント内の特定の外部キーストアには制限できないことに注意してください。

```
{
  "Sid": "AllowKeysInExternalKeyStores",
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:us-west-2:111122223333:key/*",
  "Condition": {
    "StringEquals": {
      "kms:KeyOrigin": "EXTERNAL_KEY_STORE"
    }
  }
}
```

が外部キーストアプロキシと通信 AWS KMS することを許可する

AWS KMS は、指定した外部キー[ストアプロキシを介してのみ、外部キー](#)マネージャーと通信します。AWS KMS は、指定した外部キーストアプロキシ認証情報を使用して [Signature Version 4 \(SigV4\) プロセス](#)を使用してリクエストに署名することで、プロキシに対して認証します。[外部キーストアプロキシにパブリックエンドポイント接続](#)を使用している場合、AWS KMS に追加のアクセス許可は必要ありません。

ただし、[VPC エンドポイントサービス接続](#)を使用している場合は、Amazon VPC エンドポイントサービスへのインターフェイスエンドポイントを作成する AWS KMS アクセス許可を に付与する必要があります。このアクセス許可は、外部キーストアプロキシが VPC 内にあるか、外部キーストアプロキシが他の場所にあるかにかかわらず必要ですが、 は VPC エンドポイントサービスを使用して通信します AWS KMS。

AWS KMS がインターフェイスエンドポイントを作成できるようにするには、[Amazon VPC コンソール](#)または [ModifyVpcEndpointServicePermissions](#) オペレーションを使用します。次のプリンシパルにアクセス許可を付与します: `cks.kms.<region>.amazonaws.com`。

例えば、次の AWS CLI コマンドでは、AWS KMS が米国西部 (オレゴン) (us-west-2) リージョンで指定された VPC エンドポイントサービスに接続できるようにします。このコマンドを使用する前に、Amazon VPC サービス ID と AWS リージョン を設定の有効な値に置き換えます。

```
modify-vpc-endpoint-service-permissions
--service-id vpce-svc-12abc34567def0987
--add-allowed-principals '["cks.kms.us-west-2.amazonaws.com"]'
```

このアクセス許可を削除するには、[Amazon VPC コンソール](#)か、[ModifyVpcEndpointServicePermissions](#) を `RemoveAllowedPrincipals` パラメータと共に使用します。

外部キーストアプロキシ認可 (オプション)

外部キーストアプロキシの中には、外部キーを使用するための認可要件を実装しているものがあります。外部キーストアプロキシは、特定のユーザーが特定の条件下でのみ特定のオペレーションをリクエストすることを許可する、認可スキームの設計と実装が許可されていますが、必須ではありません。例えばプロキシは、ユーザー A に、特定の外部キーを使用した暗号化は許可するが、それを使用した復号は許可しないように設計されていることがあります。

プロキシ認可は、がすべての外部キーストアプロキシに AWS KMS 必要とする [SigV4-basedプロキシ認証](#)とは無関係です。また、外部キーストアやその KMS キーに影響するオペレーションへのアクセスを認可する、キーポリシー、IAM ポリシー、グラントとも無関係です。

外部キーストアプロキシによる認可を有効にするために、は、呼び出し元、KMS キー、AWS KMS オペレーション、AWS のサービス (存在する場合) など、各[プロキシ API リクエスト](#)にメタデータ AWS KMS を含めます。外部キープロキシ API のバージョン 1 (v1) のリクエストメタデータは、次のとおりです。

```
"requestMetadata": {
  "awsPrincipalArn": string,
  "awsSourceVpc": string, // optional
  "awsSourceVpce": string, // optional
  "kmsKeyArn": string,
  "kmsOperation": string,
  "kmsRequestId": string,
  "kmsViaService": string // optional
}
```

例えば、特定のプリンシパル (awsPrincipalArn) からのリクエストを許可するようにプロキシを設定できますが、リクエストがプリンシパルに代わって特定の AWS のサービス () によって行われた場合のみですkmsViaService。

プロキシ認可に失敗すると、関連する AWS KMS オペレーションは失敗し、エラーの内容を説明するメッセージが表示されます。詳細については、「[プロキシの承認に関する問題](#)」を参照してください。

mTLS 認証 (オプション)

外部キーストアプロキシがリクエストを認証できるようにするには AWS KMS、AWS KMS は、外部キーストアの署名 V4 (SigV4) プロキシ[認証情報を使用して、外部キーストアプロキシ](#)へのすべてのリクエストに署名します。

外部キーストアプロキシが AWS KMS リクエストにのみ応答することをさらに保証するために、一部の外部キープロキシは相互 Transport Layer Security (mTLS) をサポートしています。この相互 Transport Layer Security では、トランザクションの当事者の両方が証明書を使用して相互に認証します。mTLS は、クライアント側の認証を追加します。ここで、外部キーストアプロキシサーバーは、標準 TLS が提供するサーバー側の認証に AWS KMS クライアント側の認証を追加します。万が一、プロキシ認証の認証情報が漏洩した場合、mTLS は、第三者が外部キーストアプロキシに API リクエストを実行することを防ぎます。

mTLS を実装するには、以下のプロパティを持つクライアント側の TLS 証明書のみを受け入れるように、外部キーストアプロキシを設定します。

- TLS 証明書のサブジェクトの共通名は `cks.kms.<Region>.amazonaws.com` のようにします。
例えば、`cks.kms.eu-west-3.amazonaws.com` などです。
- この証明書は、[Amazon Trust Services](#) に関連づけられた認証局に、紐付けられている必要があります。

外部キーストアプロキシとの接続オプションを選択する

外部キーストアを作成する前に、が外部キーストアコンポーネントと AWS KMS 通信する方法を決定する接続オプションを選択します。選択した接続オプションによって、残りの計画プロセスが決まります。

外部キーストアを作成する場合は、が[外部キーストアプロキシ](#)と AWS KMS 通信する方法を決定する必要があります。この選択により、必要なコンポーネントとその設定方法が決まります。は次の接続オプション AWS KMS をサポートしています。パフォーマンスとセキュリティの目標に合ったオプションを選択します。

開始する前に、[外部キーストアが必要であることを確認してください](#)。ほとんどのお客様は、キーマテリアルでバックアップされた KMS AWS KMS キーを使用できます。

Note

外部キーストアプロキシが外部キーマネージャーに組み込まれている場合は、接続が事前に決められている可能性があります。ガイダンスについては、外部キーマネージャーまたは外部キーストアプロキシのドキュメントを参照してください。

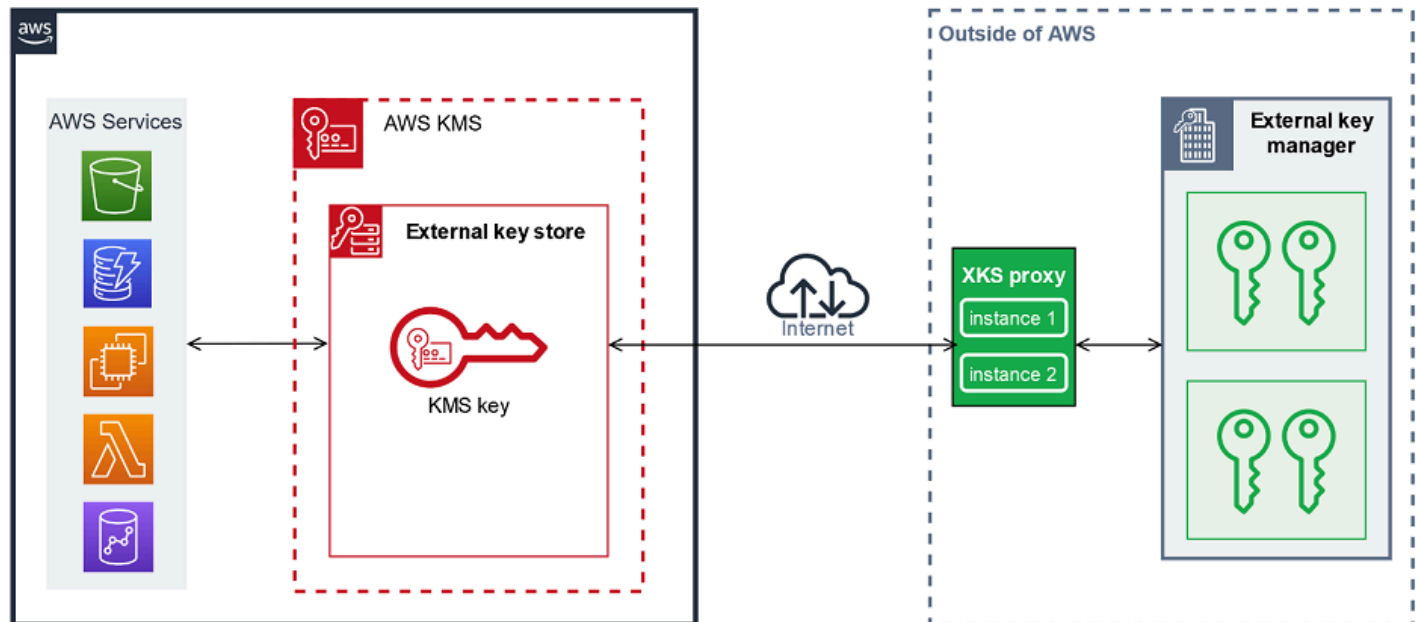
[外部キーストアプロキシの接続オプション](#)は、稼働中の外部キーストアでも変更できます。ただし、中断を最小限に抑え、エラーを回避し、データを暗号化する暗号化キーに継続的にアクセスできるように、プロセスを慎重に計画して実行する必要があります。

パブリックエンドポイント接続

AWS KMS は、パブリックエンドポイントを使用してインターネット経由で外部キーストアプロキシ (XKS プロキシ) に接続します。

この接続オプションはセットアップと保守が簡単で、一部のキー管理モデルとも問題なく連携します。ただし、一部の組織のセキュリティ要件を満たしていない場合があります。

XKS proxy connected by a public endpoint



要件

パブリックエンドポイント接続を選択する場合、以下が必要です。

- 外部キーストアプロキシは、パブリックにルーティング可能なエンドポイントからアクセスできる必要があります。
- [プロキシ URI パス](#) 値が異なる場合は、複数の外部キーストアに同じパブリックエンドポイントを使用できます。
- キーストアが異なる にもある場合でも AWS リージョン、パブリックエンドポイント接続を持つ外部キーストアと、同じ 内の VPC エンドポイントサービス接続を持つ外部キーストアに同じエンドポイントを使用することはできません AWS アカウント。
- 外部キーストアでサポートされている公開認証機関が発行した TLS 証明書を取得する必要があります。リストについては、「[Trusted Certificate Authorities](#)」(信頼された証明機関) を参照してください。

TLS 証明書のサブジェクト共通名 (CN) は、[外部キーストアプロキシのプロキシ URI エンドポイント](#) のドメイン名と一致する必要があります。例えば、パブリックエンドポイントが `https://myproxy.xks.example.com` の場合、TLS、TLS 証明書の CN は、`myproxy.xks.example.com` または `*.xks.example.com` である必要があります。

- AWS KMS と外部キーストアプロキシ間のファイアウォールで、プロキシのポート 443 との間のトラフィックが許可されていることを確認します。IPv4 経由でポート 443 で AWS KMS 通信します。この値は設定できません。

外部キーストアのすべての要件については、[前提条件を構成する](#)を参照してください。

VPC エンドポイントサービス接続

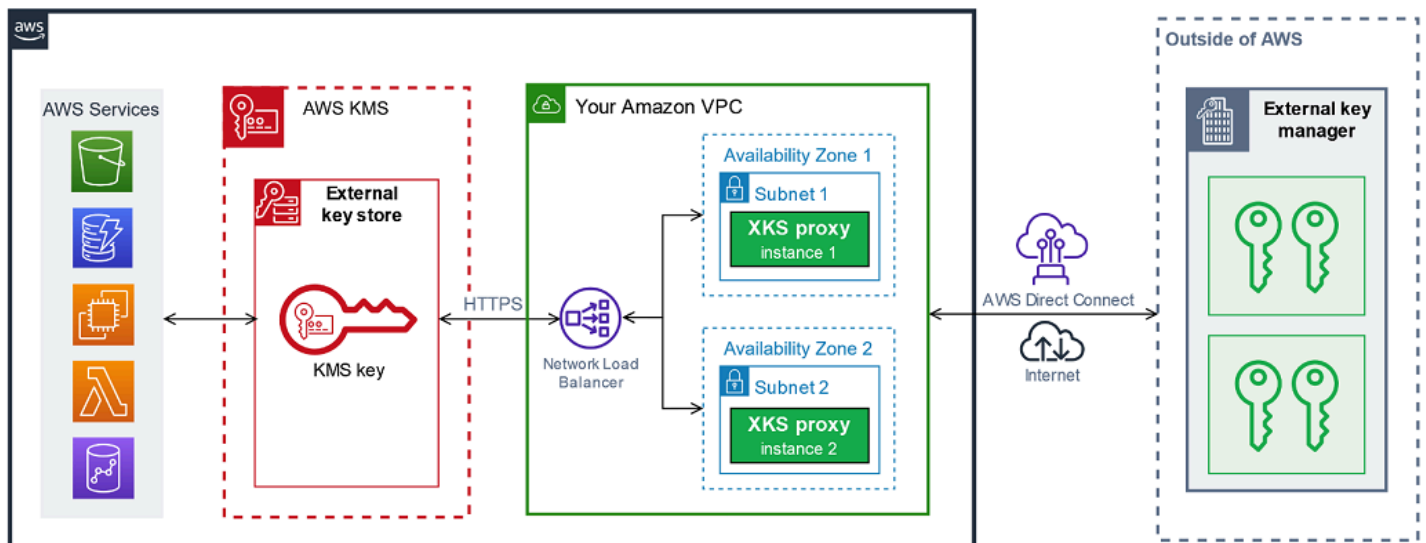
AWS KMS は、作成した Amazon VPC エンドポイントサービスへのインターフェイスエンドポイントを作成して、外部キーストアプロキシ (XKS プロキシ) に接続します。ユーザーは、[VPC エンドポイントサービス](#)を作成し、VPC を外部キーマネージャーに接続する責任があります。

エンドポイントサービスの通信には、[AWS Direct Connect](#) を含むすべての[サポートされている Network-to-Amazon VPC オプション](#)を使用できます。

この接続オプションは、セットアップと保守が複雑です。ただし AWS PrivateLink、を使用します。これにより、AWS KMS はパブリックインターネットを使用せずに Amazon VPC と外部キーストアプロキシにプライベートに接続できます。

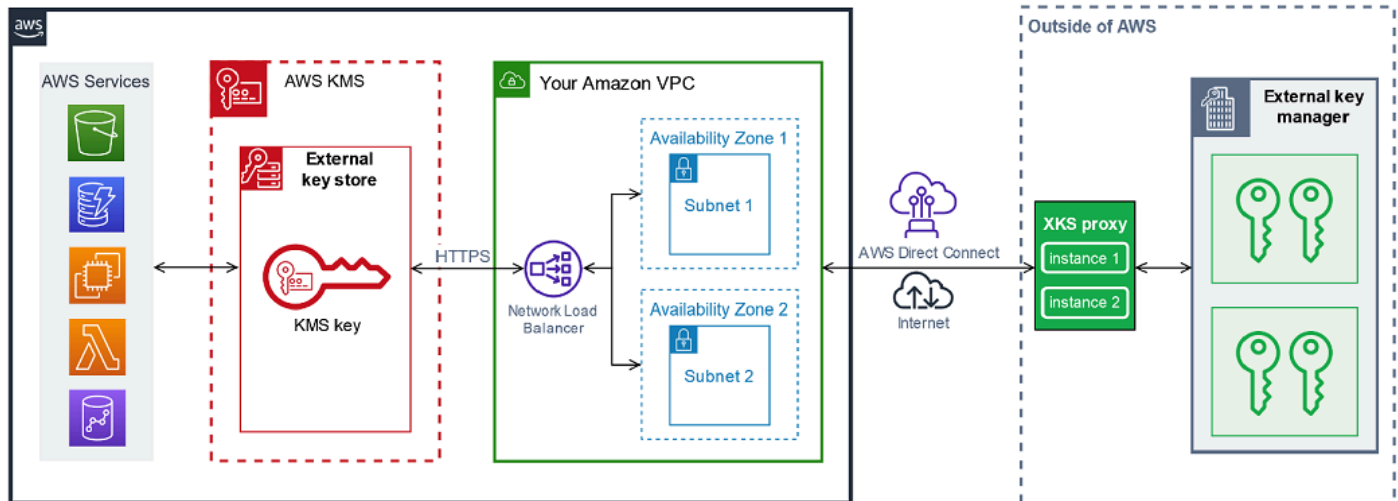
外部キーストアプロキシは Amazon VPC にあります。

XKS proxy hosted in Amazon VPC



または、外部キーストアプロキシを の外部に配置し AWS 、安全な通信のためにのみ Amazon VPC エンドポイントサービスを使用します AWS KMS。

XKS proxy connected via Amazon VPC endpoint service



詳細はこちら:

- [前提条件の組み合わせ](#)を含む、外部キーストアを作成するためのプロセスを確認します。外部キーストアを作成する際に、必要なコンポーネントがすべて揃っていることを確認するのに役立ちます。
- 外部キーストア管理者およびユーザーが必要とする許可を含む、[外部キーストアへのアクセスを制御する](#)方法について説明します。
- が外部キーストア AWS KMS に記録する [Amazon CloudWatch メトリクスとディメンション](#)について説明します。パフォーマンスや運用上の問題の兆候を早期に検出するために、外部キーストアをモニタリングするアラームを作成することを強くお勧めします。

VPC エンドポイントサービス接続を設定する

このセクションのガイダンスを使用して、[VPC エンドポイントサービス接続](#)を使用する外部キーストアに必要な AWS リソースおよび関連コンポーネントを作成および設定します。この接続オプションに一覧表示されているリソースは、[すべての外部キーストアに必要なリソース](#)を補足するものです。必要なリソースを作成、設定後に、[外部キーストアを作成](#)できます。

Amazon VPC で外部キーストアプロキシを見つけるか、の外部でプロキシを見つけ AWS、VPC エンドポイントサービスを通信に使用できます。

開始する前に、[外部キーストアが必要であることを確認してください](#)。ほとんどのお客様は、キーマテリアルでバックアップされた KMS AWS KMS キーを使用できます。

Note

VPC エンドポイントサービスの接続に必要な要素の一部は、外部キーマネージャーに含まれている場合があります。また、ソフトウェアに追加の設定要件が存在する場合があります。このセクションの AWS リソースを作成して設定する前に、プロキシとキーマネージャーのドキュメントを参照してください。

トピック

- [VPC エンドポイントサービス接続の要件](#)
- [ステップ 1: Amazon VPC とサブネットを作成する](#)
- [ステップ 2: ターゲットグループを作成する](#)
- [ステップ 3: Network Load Balancer を作成する](#)
- [ステップ 4: VPC エンドポイントサービスを作成する](#)
- [ステップ 5: プライベート DNS 名ドメインを検証する](#)
- [ステップ 6: VPC エンドポイントサービスへの接続 AWS KMS を許可する](#)

VPC エンドポイントサービス接続の要件

外部キーストアの VPC エンドポイントサービス接続を選択する場合、次のリソースが必要です。

ネットワークレイテンシーを最小限に抑えるには、[外部キーマネージャー](#)に最も近い[サポートされている AWS リージョン](#)に AWS コンポーネントを作成します。可能な場合は、ネットワークラウンドトリップ時間 (RTT) が 35 ミリ秒以下のリージョンを選択します。

- 外部キーマネージャーに接続されている Amazon VPC。2 つの異なるアベイラビリティゾーンで 2 つ以上のプライベート[サブネット](#)が必要です。

外部キーストアでの使用の[要件を満たしている](#)場合は、外部キーストアに既存の Amazon VPC を使用できます。複数の外部キーストアが Amazon VPC を共有できますが、各外部キーストアに独自の VPC エンドポイントサービスとプライベート DNS 名が必要です。

- [Network Load Balancer](#) と [ターゲットグループ](#)を備え、[AWS PrivateLinkを搭載した Amazon VPC エンドポイントサービス](#)です。

エンドポイントサービスは承認を要求できません。また、許可されたプリンシパルとして AWS KMS を追加する必要があります。これにより、AWS KMS は外部キーストアプロキシと通信できるようにインターフェイスエンドポイントを作成できます。

- AWS リージョン内で一意の、VPC エンドポイントサービスのプライベート DNS 名です。

プライベート DNS 名は、上位のパブリックドメインのサブドメインである必要があります。例えば、プライベート DNS 名が myproxy-private.xks.example.com の場合、xks.example.com または example.com などのパブリックドメインのサブドメインである必要があります。

プライベート DNS 名の DNS ドメインの[所有権を検証する](#)必要があります。

- 外部キーストアプロキシの[サポートされている公開認証機関](#)が発行した TLS 証明書。

TLS 証明書のサブジェクト共通名 (CN) は、プライベート DNS 名と一致している必要があります。例えば、プライベート DNS 名が myproxy-private.xks.example.com の場合、TLS 証明書の CN は、myproxy-private.xks.example.com または *.xks.example.com である必要があります。

外部キーストアのすべての要件については、[前提条件を構成する](#)を参照してください。

ステップ 1: Amazon VPC とサブネットを作成する

VPC エンドポイントサービス接続には、2 つ以上のプライベートサブネットをもつ外部キーマネージャーに接続された Amazon VPC が必要です。Amazon VPC を作成することも、外部キーストアの要件を満たす既存の Amazon VPC を使用することもできます。新しい Amazon VPC 作成のヘルプについては、「Amazon Virtual Private Cloud ユーザーガイド」の「[VPC を作成する](#)」を参照してください。

Amazon VPC の要件

VPC エンドポイントサービス接続を使用して外部キーストアを操作するには、Amazon VPC に次のプロパティが必要です。

- 外部キーストアと同じ AWS アカウント および[サポートされている リージョン](#)に存在する必要があります。
- それぞれ異なるアベイラビリティゾーンに、2 つ以上のプライベートサブネットが必要です。
- Amazon VPC のプライベート IP アドレス範囲が、[外部キーマネージャー](#)をホストするデータセンターのプライベート IP アドレス範囲と重複しないようにする必要があります。
- すべてのコンポーネントが IPv4 を使用する必要があります。

Amazon VPC を外部キーストアプロキシに接続するには、多数の方法があります。必要なパフォーマンスとセキュリティのニーズを満たすオプションを選択します。リストについては、「[VPC を他のネットワークに接続する](#)」および「[Network-to-Amazon VPC の接続オプション](#)」を参照してください。詳細については、「[AWS Direct Connect](#)」、および「[AWS Site-to-Site VPN ユーザーガイド](#)」を参照してください。

外部キーストア用 Amazon VPC を作成する

次の手順に従って、外部キーストア用の Amazon VPC を作成します。Amazon VPC は、[VPC エンドポイントサービス接続](#) オプションを選択した場合にのみ必要です。外部キーストアの要件を満たす既存の Amazon VPC を使用できます。

次の必須値を使用して、「[VPC、サブネット、他の VPC リソースを作成する](#)」トピックの手順に従います。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
IPv4 CIDR ブロック	VPC の IP アドレスを入力します。Amazon VPC のプライベート IP アドレス範囲が、 外部キーマネージャー をホストするデータセンターのプライベート IP アドレス範囲と重複しないようにする必要があります。
アベイラビリティゾーン数 (AZ)	2 以上
パブリックサブネット数	何も必要ありません (0)
プライベートサブネット数	AZ ごとに 1 つ
NAT ゲートウェイ	何も必要ありません。
VPC エンドポイント	何も必要ありません。
[Enable DNS hostnames]	はい

フィールド	値
(DNS ホスト名を有効化)	
DNS 解決を有効にする	はい

必ず VPC 通信をテストしてください。例えば、外部キーストアプロキシが Amazon VPC にはない場合は、Amazon VPC に Amazon EC2 インスタンスを作成し、Amazon VPC が外部キーストアプロキシと通信できることを検証します。

VPC を外部キーマネージャーに接続する

Amazon VPC がサポートする [ネットワーク接続オプション](#) のいずれかを使用して、外部キーマネージャーをホストするデータセンターに VPC を接続します。VPC 内の Amazon EC2 インスタンス (または、VPC 内にある場合は外部キーストアプロキシ) がデータセンターおよび外部キーマネージャーと通信できることを確認します。

ステップ 2: ターゲットグループを作成する

必要な VPC エンドポイントサービスを作成する前に、必要なコンポーネント、Network Load Balancer (NLB)、ターゲットグループを作成します。Network Load Balancer (NLB) は、リクエストを複数の正常なターゲットに分散し、いずれのターゲットもリクエストを処理できるようにします。このステップでは、外部キーストアプロキシ用に 2 つ以上のホストを含むターゲットグループを作成し、そのターゲットグループに IP アドレスを登録します。

次の必須値を使用して、「[ターゲットグループの設定](#)」トピックの手順に従います。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
対象タイプ	IP アドレス
プロトコル	TCP
ポート	443

フィールド	値
IP アドレスタイプ	IPv4
VPC	外部キーストアの VPC エンドポイントサービスを作成する VPC を選択します。
ヘルスチェックプロトコルとパス	ヘルスチェックプロトコルとパスは、外部キーストアプロキシの設定に応じて異なります。外部キーマネージャーまたは外部キーストアプロキシのドキュメントを参照してください。 ターゲットグループのヘルスチェック設定に関する一般情報は、Elastic Load Balancing ユーザーガイドの Network Load Balancer の項目にある「 ターゲットグループのヘルスチェック 」を参照してください。
ネットワーク	その他のプライベート IP アドレス
IPv4 アドレス	外部キーストアプロキシのプライベートアドレス
ポート	443

ステップ 3: Network Load Balancer を作成する

Network Load Balancer は、外部キーストアプロキシへの AWS KMS からのリクエストなどのネットワークトラフィックを、設定されたターゲットに分散します。

「[ロードバランサーとリスナーの設定](#)」トピックの手順に従ってリスナーを設定および追加し、次の必須値を使用してロードバランサーを作成します。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
スキーム	内部
IP アドレスタイプ	IPv4
ネットワークマッピング	外部キーストアの VPC エンドポイントサービスを作成する VPC を選択します。

フィールド	値
マッピング	VPC サブネットに設定した両方のアベイラビリティーゾーン (2 つ以上) を選択します。サブネット名とプライベート IP アドレスを検証します。
プロトコル	TCP
ポート	443
デフォルトアクション: 転送	Network Load Balancer の ターゲットグループ を選択します。

ステップ 4: VPC エンドポイントサービスを作成する

通常、サービスへのエンドポイントを作成します。ただし、VPC エンドポイントサービスを作成すると、ユーザーはプロバイダーになり、サービスへのエンドポイント AWS KMS が作成されます。外部キーストアの場合は、前のステップで作成した Network Load Balancer で VPC エンドポイントサービスを作成します。VPC エンドポイントサービスは、外部キーストアと同じ AWS アカウントサポートされている [リージョン](#) に存在する必要があります。

複数の外部キーストアが Amazon VPC を共有できますが、各外部キーストアに独自の VPC エンドポイントサービスとプライベート DNS 名が必要です。

「[エンドポイントサービスの作成](#)」トピックの手順に従って、次の必須値を含む VPC エンドポイントサービスを作成します。他のフィールドについては、デフォルト値を受け入れ、必要に応じて名前を指定します。

フィールド	値
ロードバランサーのタイプ	ネットワーク
使用可能なロードバランサー	前のステップで作成した Network Load Balancer を選択します。 新しいロードバランサーがリストに表示されない場合は、その状態がアクティブであることを確認します。ロードバランサーの状態がプロビジョニングからアクティブに変わるまでに、数分かかる場合があります。
承認が必要です	False。チェックボックスをオフにします。

フィールド	値
	acceptance. AWS KMS can not connect to the VPC endpoint service without a manual acceptance. 承認が必要な場合、 外部キーストアを作成 しようとする と、XksProxyInvalidConfigurationException の例外が発生して 失敗します。
プライベート DNS 名を有効 にする	プライベート DNS 名をサービスに関連付ける
プライベート DNS 名	AWS リージョンで一意的プライベート DNS 名を入力します。 プライベート DNS 名は、上位のパブリックドメインのサブドメインである 必要があります。例えば、プライベート DNS 名が myproxy-private.xk s.example.com の場合、xks.example.com または example.com な どのパブリックドメインのサブドメインである必要があります。 このプライベート DNS 名は、外部キーストアプロキシに設定されている TLS 証明書のサブジェクト共通名 (CN) と一致する必要があります。例えば、 プライベート DNS 名が myproxy-private.xks.example.com の場 合、TLS 証明書の CN は、myproxy-private.xks.example.com また は *.xks.example.com である必要があります。 証明書とプライベート DNS 名が一致しない場合、外部キーストアを外 部キーストアプロキシに接続しようすると、XKS_PROXY_INVALID_ TLS_CONFIGURATION の接続エラーコードが発生して失敗します。詳細に ついては、「一般的な設定エラー」を参照してください。
サポートされて いる IP アドレ スのタイプ	IPv4

ステップ 5: プライベート DNS 名ドメインを検証する

VPC エンドポイントサービスを作成すると、そのドメイン検証ステータスは
pendingVerification になります。VPC エンドポイントサービスを使用して外部キーストアを
作成する前に、このステータスが verified になっている必要があります。プライベート DNS 名

に関連付けられたドメインを所有していることを検証するには、パブリック DNS サーバーに TXT レコードを作成する必要があります。

たとえば、VPC エンドポイントサービスのプライベート DNS 名が `myproxy-private.xks.example.com`、`xks.example.com` または `example.com` などのパブリックドメインに TXT レコードを作成する必要があります。は最初に `example.com` で TXT レコードを探し、次に `xks.example.com` で TXT レコード AWS PrivateLink を探します。

 Tip

TXT レコードを追加した後、[Domain verification status] (ドメイン検証ステータス) の値が `pendingVerification` から `verify` に変わるまでに数分かかる場合があります。

最初に、次のいずれかの方法でドメインの検証ステータスを確認します。有効な値は、`verified`、`pendingVerification`、`failed` です。

- [Amazon VPC コンソール](#)で、[Endpoint services] (エンドポイントサービス) を選択し、エンドポイントサービスを選択します。詳細ペインで、[Domain verification status] (ドメイン検証ステータス) を参照してください。
- [DescribeVpcEndpointServiceConfigurations](#) オペレーションを使用します。State 値は `ServiceConfigurations.PrivateDnsNameConfiguration.State` フィールドにあります。

検証ステータスが `verified` でない場合は、[ドメイン所有権の検証](#) トピックの手順に従ってドメインの DNS サーバーに TXT レコードを追加し、TXT レコードが公開されていることを検証します。次に、検証ステータスを再度チェックします。

プライベート DNS ドメイン名の A レコードを作成する必要はありません。が VPC エンドポイントサービスへのインターフェイスエンドポイント AWS KMS を作成すると、AWS PrivateLink は AWS KMS VPC 内のプライベートドメイン名に必要な A レコードを持つホストゾーンを自動的に作成します。これは、VPC エンドポイントサービス接続を備える外部キーストアの場合、[外部キーストア](#) を外部キーストアプロキシに接続したときに発生します。

ステップ 6: VPC エンドポイントサービスへの接続 AWS KMS を に許可する

VPC エンドポイントサービスのプリンシパルを許可するリスト AWS KMS に を追加する必要があります。これにより、AWS KMS は VPC エンドポイントサービスへのインターフェイスエンドポイント

トを作成できます。AWS KMS が許可されたプリンシパルでない場合、外部キーストアの作成の試行は `XksProxyVpcEndpointServiceNotFoundException` 例外で失敗します。

「AWS PrivateLink ガイド」の「[許可を管理する](#)」トピックの手順に従ってください。次の必須値を使用します。

フィールド	値
ARN	<code>cks.kms.<region>.amazonaws.com</code> 例: <code>cks.kms.us-east-1.amazonaws.com</code>

Next: [外部キーストアを作成する](#)

外部キーストアを作成する

各 AWS アカウント およびリージョンに 1 つまたは複数の外部キーストアを作成できます。各外部キーストアは AWS、 の外部キーマネージャー、および AWS KMS と外部キーマネージャー間の通信を仲介する外部キーストアプロキシ (XKS プロキシ) に関連付ける必要があります。詳細については、「[外部キーストアプロキシとの接続オプションを選択する](#)」を参照してください。開始する前に、[外部キーストアが必要であることを確認してください](#)。ほとんどのお客様は、キーマテリアルでバックアップされた KMS AWS KMS キーを使用できます。

Tip

外部キーマネージャーの中には、外部 キーを作成するための簡単な方法が用意されているものもあります。詳細については外部キーマネージャーのドキュメントを参照してください。

外部キーストアを作成するときは、事前に[前提条件を構成する](#)必要があります。作成中に、外部キーストアのプロパティを指定します。最も重要なのは、 の外部キーストアが[パブリックエンドポイント](#)を使用するか、[VPC エンドポイントサービス](#) AWS KMS を使用して外部キーストアプロキシに接続するかを示します。また、プロキシの URI エンドポイントや、 が API リクエストをプロキシ AWS KMS に送信するプロキシエンドポイント内のパスなど、接続の詳細も指定します。

考慮事項

- KMS は IPv6 経由で外部キーストアと通信できません。

- パブリックエンドポイント接続を使用する場合は、AWS KMS が HTTPS 接続を使用してインターネット経由でプロキシと通信できることを確認してください。これには、外部キーストアプロキシで TLS を設定し、AWS KMS とプロキシ間のファイアウォールが、プロキシのポート 443 との間の IPv4 トラフィックを許可することが含まれます。パブリックエンドポイント接続を使用して外部キーストアを作成するときに、は外部キーストアプロキシにステータスリクエストを送信して接続を AWS KMS テストします。このテストでは、エンドポイントが到達可能であることと、外部キーストアプロキシが[外部キーストアプロキシ認証の認証情報](#)で署名されたリクエストを受け入れることを、確認します。このテストリクエストに失敗すると、外部キーストアを作成するオペレーションは失敗します。
- VPC エンドポイントサービス接続を使用する場合は、Network Load Balancer、プライベート DNS 名、VPC エンドポイントサービスが正しく設定され動作していることを確認します。外部キーストアプロキシが VPC にない場合は、VPC エンドポイントサービスが外部キーストアプロキシと通信できることを確認する必要があります ([外部キーストアを外部キーストアプロキシに接続する](#)と、VPC エンドポイントサービスの接続を AWS KMS テストします)。
- AWS KMS は、特に外部キーストアの [Amazon CloudWatch メトリクスとディメンション](#)を記録します。これらのメトリクスの一部に基づくモニタリンググラフは、外部キーストアごとに AWS KMS コンソールに表示されます。これらのメトリクスを使用して、外部キーストアをモニタリングするアラームを作成することを強くお勧めします。これらのアラームは、パフォーマンスやオペレーションに関する問題の初期兆候を、発生前にユーザーに警告します。手順については、「[外部キーストアをモニタリングする](#)」を参照してください。
- 外部キーストアは[リソースクォータ](#)の影響を受けます。外部キーストアで KMS キーを使用すると、[リクエストクォータ](#)の影響を受けます。外部キーストアの実装を設計するときは、事前にこれらのクォータを確認します。

Note

動作を妨げる可能性のある依存関係の循環がないか、設定を確認します。

たとえば、AWS リソースを使用して外部キーストアプロキシを作成する場合は、プロキシを操作する際に、そのプロキシを介してアクセスされる外部キーストアで KMS キーを使用できる必要がないことを確認してください。

すべての新しい外部キーストアは、切断された状態で作成されます。外部キーストアに KMS キーを作成するときは、外部キーストアを外部キーストアプロキシに[接続しておく](#)必要があります。外部キーストアのプロパティを変更するには、[外部キーストアの設定を編集](#)します。

トピック

- [前提条件を構成する](#)
- [新しい外部キーストアを作成する](#)

前提条件を構成する

外部キーストアを作成する前に、外部キーストアのサポートに使用する[外部キーマネージャー](#)や、リクエストを[外部キーマネージャーが理解できる形式に変換する外部キーストアプロキシ](#)など、必要なコンポーネントをアセンブルする必要があります。AWS KMS

以下のコンポーネントは、すべての外部キーストアに必要です。これらのコンポーネントの他に、選択した[外部キーストアプロキシ接続オプション](#)をサポートするコンポーネントを、指定する必要があります。

Tip

外部キーマネージャーにこれらのコンポーネントの一部が含まれているか、あるいはそれらが自動的に構成されている場合があります。詳細については外部キーマネージャーのドキュメントを参照してください。

AWS KMS コンソールで外部キーストアを作成する場合は、プロキシ [URI パスとプロキシ認証情報](#)を指定する JSON ベースの[プロキシ設定ファイル](#)をアップロードできます。一部の外部キーストアプロキシでは、このファイルは自動的に生成されます。詳細については、外部キーストアプロキシか外部キーマネージャーのドキュメントを参照してください。

外部キーマネージャー

各外部キーストアには 1 つ以上の[外部キーマネージャー](#)インスタンスが必要です。これには、物理的または仮想的なハードウェアセキュリティモジュール (HSM)、またはキー管理ソフトウェアを使用できます。

キーマネージャーは 1 つだけ使用できますが、冗長性を確保するために、暗号化キーを共有する、関連付けられたキーマネージャーインスタンスを 2 つ以上用意しておくことが推奨されます。外部キーストアでは、外部キーマネージャーを独占的に使用する必要はありません。ただし、外部キーマネージャーには、リソースを保護するために外部キーストアで KMS キーを使用する AWS サービスからの暗号化および復号リクエストの予想される頻度を処理する容量が必要です。外部キーマネージャーは、1 秒あたり最大 1,800 件のリクエストを処理し、各リクエストで 250 ミリ秒のタイムア

ウト内に応答するように設定する必要があります。ネットワーク往復時間 (RTT) が 35 ミリ秒以下 AWS リージョン になるように、外部キーマネージャーを の近くに配置することをお勧めします。

外部キーストアプロキシで許可されている場合は、外部キーストアプロキシに関連付ける外部キーマネージャーを変更できますが、新しい外部キーマネージャーは、同じキーマテリアルを持つバックアップまたはスナップショットでなければなりません。KMS キーに関連付ける外部キーが外部キーストアプロキシで使用できなくなった場合、AWS KMS は KMS キーで暗号化された暗号文を復号できません。

外部キーマネージャーは、外部キーストアプロキシからアクセスできなければなりません。プロキシからの [GetHealthStatus](#) レスポンスですべての外部キーマネージャーインスタンスが Unavailable であると報告された場合、外部キーストアを作成しようとする [XksProxyUriUnreachableException](#) によりすべて失敗します。

外部キーストアプロキシ

ユーザーは、[AWS KMS 外部キーストアプロキシ API 仕様](#)の設計要件に従う[外部キーストアプロキシ](#) (XKS プロキシ) を指定する必要があります。外部キーストアプロキシを開発または購入するか、によって提供される、または外部キーマネージャーに組み込まれた外部キーストアプロキシを使用できます。AWS KMS は、外部キーストアプロキシを 1 秒あたり最大 1,800 件のリクエストを処理し、各リクエストの 250 ミリ秒のタイムアウト内に応答するように設定することをお勧めします。ネットワーク往復時間 (RTT) が 35 ミリ秒以下 AWS リージョン になるように、外部キーマネージャーを の近くに配置することをお勧めします。

外部キーストアプロキシは複数の外部キーストアに使用できますが、各外部キーストアには、固有の URI エンドポイントと、リクエストに対応する外部キーストアプロキシ内のパスが必要です。

VPC エンドポイントサービス接続を使用している場合、Amazon VPC に外部キーストアプロキシを配置できますが必須ではありません。プライベートデータセンターなど AWS、の外部でプロキシを見つけ、VPC エンドポイントサービスはプロキシとの通信にのみ使用できます。

プロキシ認証の認証情報

外部キーストアを作成するときは、外部キーストアプロキシ認証の認証情報 (XksProxyAuthenticationCredential) を指定する必要があります。

外部キーストアプロキシ AWS KMS で [認証情報](#) (XksProxyAuthenticationCredential) を確立する必要があります。AWS KMS は、外部キーストアプロキシ認証情報を使用して [署名バージョン 4 \(SigV4\) プロセス](#)を使用してリクエストに署名することで、プロキシを認証します。外部キーストアを作成するときに認証情報を指定します。これはいつでも[変更できます](#)。プロキシが認証情報をローテーションする場合は、外部キーストアの認証情報値を更新する必要があります。

プロキシ認証の認証情報は、2つの要素から成ります。外部キーストアでは両方の要素を指定する必要があります。

- アクセスキー ID: シークレットアクセスキーを識別します。この ID はプレーンテキストで入力できます。
- シークレットアクセスキー: 認証情報のシークレット部分。は、認証情報を保存する前に、認証情報内のシークレットアクセスキーを AWS KMS 暗号化します。

AWS KMS が外部キーストアプロキシへのリクエストに署名するために使用する SigV4 認証情報は、AWS アカウントのプリンシパルに関連付けられた SigV4 AWS Identity and Access Management 認証情報とは無関係です。IAM SigV4 認証情報を外部キーストアプロキシに再利用しないでください。

プロキシ接続

外部キーストアを作成するときは、外部キーストアプロキシ認証の接続オプション (XksProxyConnectivity) を指定する必要があります。

AWS KMS は、[パブリックエンドポイント](#)または [Amazon Virtual Private Cloud \(Amazon VPC\) エンドポイントサービス](#)を使用して、外部キーストアプロキシと通信できます。パブリックエンドポイントは設定と保守が容易ですが、あらゆるインストールのセキュリティ要件を満たしているとは限りません。Amazon VPC エンドポイントサービスの接続オプションを選択する場合は、必要なコンポーネント (2つの異なるアベイラビリティゾーンに2つ以上のサブネットがある Amazon VPC、ネットワークロードバランサーとターゲットグループを含む VPC エンドポイントサービス、VPC エンドポイントサービスのプライベート DNS 名など) を作成し、管理する必要があります。

外部キーストアの[プロキシ接続オプションは変更できます](#)。ただし、外部キーストアの、KMS キーに関連付けられたキーマテリアルが、引き続き利用可能であることを確認する必要があります。それ以外の場合、AWS KMS はそれらの KMS キーで暗号化された暗号文を復号できません。

外部キーストアに最適なプロキシ接続オプションを決める方法については、[外部キーストアプロキシとの接続オプションを選択する](#)を参照してください。VPC エンドポイントサービス接続の作成と設定に関するヘルプは、[VPC エンドポイントサービス接続を設定する](#)を参照してください。

プロキシ URI エンドポイント

外部キーストアを作成するには、AWS KMS が外部キーストアプロキシにリクエストを送信するために使用するエンドポイント (XksProxyUriEndpoint) を指定する必要があります。

プロトコルは、ポート 443 の IPv4 経由で HTTPS. AWS KMS communicates である必要があります。プロキシ URI エンドポイント値のポートを指定しないようにします。

- [パブリックエンドポイント接続](#) — 外部キーストアプロキシ用に、公開されているエンドポイントを指定します。このエンドポイントには、外部キーストアを作成する前にアクセスできる必要があります。
- [VPC エンドポイントサービス接続](#) — 後に VPC エンドポイントサービスのプライベート DNS 名が続く `https://` を指定します。

外部キーストアプロキシに設定された TLS サーバ証明書は、外部キーストアプロキシ URI エンドポイントのドメイン名と一致し、外部キーストアでサポートされた認証局が発行している必要があります。リストについては、「[Trusted Certificate Authorities](#)」(信頼された証明機関) を参照してください。証明機関は、TLS 証明書を発行する前に、ドメイン所有権の証明を要求します。

TLS 証明書のサブジェクト共通名 (CN) は、プライベート DNS 名と一致している必要があります。例えば、プライベート DNS 名が `myproxy-private.xks.example.com` の場合、TLS 証明書の CN は、`myproxy-private.xks.example.com` または `*.xks.example.com` である必要があります。

[プロキシ URI エンドポイント](#) は変更が可能ですが、必ず、外部キーストアプロキシが、外部キーストアの KMS キーに関連付けられたキーマテリアルにアクセスできることを確認します。それ以外の場合、AWS KMS はそれらの KMS キーで暗号化された暗号文を復号できません。

一意性の要件

- プロキシ URI エンドポイント (`XksProxyUriEndpoint`) とプロキシ URI パス (`XksProxyUriPath`) を組み合わせた値は、AWS アカウント およびリージョン内で、一意でなければなりません。
- パブリックエンドポイントに接続可能な外部キーストアは、プロキシ URI パス値が異なっている限り、同じプロキシ URI エンドポイントを共有できます。
- パブリックエンドポイント接続を持つ外部キーストアは、キーストアが異なる場合でも AWS リージョン、同じに VPC エンドポイントサービス接続を持つ外部キーストアと同じプロキシ URI エンドポイント値を使用することはできません AWS アカウント。
- VPC エンドポイントに接続可能な各外部キーストアには、それぞれ独自のプライベート DNS 名が必要です。プロキシ URI エンドポイント (プライベート DNS 名) は、AWS アカウント およびリージョンで一意である必要があります。

プロキシ URI パス

外部キーストアを作成するには、外部キーストアプロキシで、[必要なプロキシ API](#) へのベースパスを指定する必要があります。値は / から始まり、/kms/xks/v1 で終わる必要があります。v1 は、外部キーストアプロキシの AWS KMS API のバージョンを表しています。このパスでは、必須の要素の間に、/example-prefix/kms/xks/v1 のようなプレフィクスをオプションで含めることができます。この値は、外部キーストアプロキシのドキュメントでご確認いただけます。

AWS KMS は、プロキシ URI エンドポイントとプロキシ URI パスの連結で指定されたアドレスにプロキシリクエストを送信します。たとえば、プロキシ URI エンドポイントが `https://myproxy.xks.example.com` で、プロキシ URI パスが `/kms/xks/v1` の場合、はプロキシ API リクエストを AWS KMS に送信します `https://myproxy.xks.example.com/kms/xks/v1`。

[プロキシ URI パス](#) は変更が可能ですが、必ず、外部キーストアプロキシが、外部キーストアの KMS キーに関連付けられたキーマテリアルにアクセスできることを確認します。それ以外の場合、AWS KMS はそれらの KMS キーで暗号化された暗号文を復号できません。

一意性の要件

- プロキシ URI エンドポイント (XksProxyUriEndpoint) とプロキシ URI パス (XksProxyUriPath) を組み合わせた値は、AWS アカウント およびリージョン内で、一意でなければなりません。

VPC エンドポイントサービス

外部キーストアプロキシとの通信に使用する Amazon VPC エンドポイントサービスの名前を指定します。このコンポーネントが必要なのは、VPC エンドポイントサービス接続を使用する外部キーストアのみです。外部キーストア向け VPC エンドポイントサービスの、セットアップと設定方法のヘルプについては、[VPC エンドポイントサービス接続を設定する](#) を参照してください。

VPC エンドポイントサービスは、以下のプロパティを持つ必要があります。

- VPC エンドポイントサービスは、外部キーストアと同じ AWS アカウント およびリージョンにある必要があります。
- Network Load Balancer (NLB) は、それぞれが異なるアベイラビリティゾーンに存在する 2 つ以上のサブネットに接続されていること。
- VPC エンドポイントサービスの許可プリンシパルリストには `cks.kms.<region>.amazonaws.com`、などのリージョン AWS KMS のサービスプリンシパルが含まれている必要があります `cks.kms.us-east-1.amazonaws.com`。

- 接続リクエストの承認を要求しないこと。
- 上位レベルのパブリックドメインにプライベート DNS 名があること。例えば、パブリック `xks.example.com` ドメインに `myproxy-private.xks.example.com` というプライベート DNS 名を設定できる。

VPC エンドポイントサービスに接続可能な、外部キーストアのプライベート DNS 名は、その AWS リージョンで一意的であること。

- プライベート DNS 名ドメインの [ドメイン検証ステータス](#) は、verified であること。
- 外部キーストアプロキシで設定された TLS サーバ証明書で、エンドポイントに到達可能なプライベート DNS ホスト名が指定されていること。

一意性の要件

- VPC エンドポイントに接続可能な外部キーストアは Amazon VPC を共有することができますが、各外部キーストアには独自の VPC エンドポイントサービスとプライベート DNS 名が必要です。

プロキシ設定ファイル

プロキシ設定ファイルは、外部キーストアの [プロキシ URI パス](#) と [プロキシ認証の認証情報](#) プロパティの値を含む、オプションの JSON ベースファイルです。AWS KMS コンソールで外部キーストアを作成または [編集](#) する場合は、プロキシ設定ファイルをアップロードすることで、外部キーストアの設定値を指定できます。このファイルを使用すれば入力や貼り付けのエラーを回避でき、外部キーストアの値を外部キーストアプロキシの値と確実に一致させることができます。

プロキシ設定ファイルは、外部キーストアプロキシによって生成されます。外部キーストアプロキシでプロキシ設定ファイルが提供されているかどうかは、外部キーストアプロキシのドキュメントでご確認いただけます。

以下は、正しい形式の Proxy 設定ファイルの例です。使用している値は架空のものです。

```
{
  "XksProxyUriPath": "/example-prefix/kms/xks/v1",
  "XksProxyAuthenticationCredential": {
    "AccessKeyId": "ABCDE12345670EXAMPLE",
    "RawSecretAccessKey": "0000EXAMPLEFA5FT0mCc3DrGUe2sti527BitkQ0Zr9M09+vE="
  }
}
```

プロキシ設定ファイルは、AWS KMS コンソールで外部キーストアを作成または編集する場合にのみアップロードできます。[CreateCustomKeyStore](#) または [UpdateCustomKeyStore](#) オペレーションでは使用できませんが、パラメータ値が正しいかどうかは、プロキシ設定ファイルの値を使用することで確認できます。

新しい外部キーストアを作成する

必要な前提条件をアセンブルしたら、AWS KMS コンソールで、または [CreateCustomKeyStore](#) オペレーションを使用して、新しい外部キーストアを作成できます。

AWS KMS コンソールの使用

外部キーストアを作成する際には、[お使いのプロキシの接続タイプを選択](#)し、すべての[必要なコンポーネント](#)が作成済み・設定済みであることをあらかじめ確認してください。必要な値を見つける際に不明なことがあれば、外部キーストアプロキシまたはキー管理ソフトウェアのドキュメントを参照してください。

Note

で外部キーストアを作成すると AWS Management Console、プロキシ [URI パスとプロキシ認証情報](#)の値を含む JSON ベースのプロキシ設定ファイルをアップロードできます。一部のプロキシでは、このファイルは自動的に生成されます。この値は必須ではありません。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[Custom key stores] (カスタムキーストア)、[External key stores] (外部キーストア) の順に選択します。
4. [Create external key store] (外部キーストアの作成) を選択します。
5. カスタムキーストア用のフレンドリ名を入力します。名前は、アカウント内のすべての外部キーストアの中で、一意でなければなりません。

Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

6. [プロキシの接続](#)タイプを選択します。

選択した接続によって、外部キーストアプロキシに[必要なコンポーネント](#)が決まります。選択方法の詳細については、[外部キーストアプロキシとの接続オプションを選択する](#)を参照してください。

- この外部キーストアの、[VPC エンドポイントサービス](#)の名前を選択または入力します。このステップは、外部キーストアプロキシの接続タイプが [VPC endpoint service] (VPC エンドポイントサービス) である場合のみ表示されます。

VPC エンドポイントサービスとその VPC は、外部キーストアの要件を満たしている必要があります。詳細については、「[the section called “前提条件を構成する”](#)」を参照してください。

- [プロキシ URI エンドポイント](#)を入力します。プロトコルは、ポート 443 の IPv4 経由で HTTPS。AWS KMS communicates である必要があります。プロキシ URI エンドポイント値のポートを指定しないようにします。

が前のステップで指定した VPC エンドポイントサービス AWS KMS を認識すると、このフィールドが自動的に完了します。

パブリックエンドポイント接続の場合は、公開されているエンドポイント URI を入力します。VPC エンドポイントサービス接続では、後ろに VPC エンドポイントサービスのプライベート DNS 名が続く https:// を入力します。

- [プロキシ URI パス](#)プレフィクスと[プロキシ認証の認証情報](#)の値を入力するには、プロキシの設定ファイルをアップロードするか、値を手動で入力します。

- [プロキシ URI パス](#)と[プロキシ認証の認証情報](#)の値を含む、オプションの[プロキシ設定ファイル](#)がある場合は、[Upload configuration file] (設定ファイルをアップロード) を選択します。ステップに従ってファイルをアップロードします。

ファイルがアップロードされると、コンソールの編集可能なフィールドに、ファイルの値が表示されます。ここで値を変更できますが、外部キーストアを作成した後でも[これらの値を編集](#)できます。

シークレットアクセスキーの値を表示するには、[Show secret access key] (シークレットアクセスキーを表示) を選択します。

- プロキシ設定ファイルがなければ、プロキシ URI パスとプロキシ認証の認証情報の値を手動で入力できます。
 - プロキシ設定ファイルがなければ、プロキシ URI を手動で入力できます。コンソールに、必要な [/kms/xks/v1] 値が入力されます。

[プロキシ URI パス](#)に、`/example-prefix/kms/xks/v1` の `example-prefix` など、オプションのプレフィクスが含まれている場合は、そのプレフィクスを [Proxy URI path prefix] (プロキシ URI パスプレフィクス) フィールドに入力します。含まれていなければ、フィールドは空のままにします。

- b. プロキシ設定ファイルがなければ、[プロキシ認証の認証情報](#)を手動で入力します。アクセスキー ID とシークレットアクセスキーの両方が必要です。
- [Proxy credential: Access key ID] (プロキシ認証情報: アクセスキー ID) に、プロキシ認証の認証情報のアクセスキー ID を入力します。アクセスキー ID は、シークレットアクセスキーを識別します。
 - [Proxy credential: Secret access key] (プロキシ認証情報: シークレットアクセスキー) に、プロキシ認証の認証情報の、シークレットアクセスキーを入力します。

シークレットアクセスキーの値を表示するには、[Show secret access key] (シークレットアクセスキーを表示) を選択します。

この手順では、外部キーストアプロキシで作成した認証情報を、設定または変更することはできません。これらの値を、外部キーストアに関連付けるだけです。プロキシ認証の認証情報を設定、変更、ローテーションする方法については、外部キーストアプロキシまたはキー管理ソフトウェアのドキュメントを参照してください。

プロキシ認証の認証情報が変更された場合は、外部キーストアの[認証情報の設定を編集](#)します。

10. [Create external key store] (外部キーストアの作成) を選択します。

手順が完了すると、アカウントとリージョンの外部キーストアのリストに、新しい外部キーストアが表示されます。正常に完了しなかった場合は、問題を説明し、修正方法を示すエラーメッセージが表示されます。さらにヘルプが必要な場合は、「[外部キーの CreateKey エラー](#)」を参照してください。

次の手順: 新しい外部キーストアが自動で接続されない。外部キーストア AWS KMS keys で を作成する前に、[外部キーストアを外部キーストアプロキシに接続](#)する必要があります。

AWS KMS API の使用

新しい外部キーストアを作成するには、[CreateCustomKeyStore](#) オペレーションを使用します。必要なパラメータの値を見つける際に不明なことがあれば、外部キーストアプロキシまたはキー管理ソフトウェアのドキュメントを参照してください。

Tip

CreateCustomKeyStore オペレーションを使用しているときは、[プロキシ設定ファイル](#)はアップロードできません。ただし、プロキシ設定ファイルの値を使って、パラメータ値が正しいことを確認することは可能です。

外部キーストアを作成するには、CreateCustomKeyStore オペレーションでは以下のパラメータ値が必要になります。

- CustomKeyStoreName — アカウント内で一意である外部キーストアのフレンドリ名。

Important

このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。

- CustomKeyStoreType - EXTERNAL_KEY_STORE を指定します。
- [XksProxyConnectivity](#) - PUBLIC_ENDPOINT または VPC_ENDPOINT_SERVICE を指定します。
- [XksProxyAuthenticationCredential](#) — アクセスキー ID とシークレットアクセスキーの両方を指定します。
- [XksProxyUriEndpoint](#) — が外部キーストアプロキシとの通信 AWS KMS に使用するエンドポイント。
- [XksProxyUriPath](#) — プロキシ内でのプロキシ API へのパスです。
- [XksProxyVpcEndpointServiceName](#) — XksProxyConnectivity 値が VPC_ENDPOINT_SERVICE の場合のみ必要です。

Note

AWS CLI バージョン 1.0 を使用する場合は、パラメータなどの HTTP または HTTPS 値を持つ `XksProxyUriEndpoint` パラメータを指定する前に、次のコマンドを実行します。

```
aws configure set cli_follow_urlparam false
```

それ以外の場合、AWS CLI バージョン 1.0 はパラメータ値をその URI アドレスにあるコンテンツに置き換え、次のエラーが発生します。

```
Error parsing parameter '--xks-proxy-uri-endpoint': Unable to retrieve
https:// : received non 200 status code of 404
```

以下の例の値は架空の値です。コマンドを実行する前に、外部キーストアに有効な値に置き換えてください。

パブリックエンドポイント接続を使用して外部キーストアを作成します。

```
$ aws kms create-custom-key-store
  --custom-key-store-name ExampleExternalKeyStorePublic \
  --custom-key-store-type EXTERNAL_KEY_STORE \
  --xks-proxy-connectivity PUBLIC_ENDPOINT \
  --xks-proxy-uri-endpoint https://myproxy.xks.example.com \
  --xks-proxy-uri-path /kms/xks/v1 \
  --xks-proxy-authentication-credential
AccessKeyId=<value>,RawSecretAccessKey=<value>
```

VPC エンドポイントサービス接続を使用して外部キーストアを作成します。

```
$ aws kms create-custom-key-store
  --custom-key-store-name ExampleExternalKeyStoreVPC \
  --custom-key-store-type EXTERNAL_KEY_STORE \
  --xks-proxy-connectivity VPC_ENDPOINT_SERVICE \
  --xks-proxy-vpc-endpoint-service-name com.amazonaws.vpce.us-east-1.vpce-svc-example \
  --xks-proxy-uri-endpoint https://myproxy-private.xks.example.com \
  --xks-proxy-uri-path /kms/xks/v1 \
  --xks-proxy-authentication-credential
AccessKeyId=<value>,RawSecretAccessKey=<value>
```


オペレーションが正常に終了したら、次のレスポンス例に示すように、CreateCustomKeyStore はカスタムキーストア ID を返します。

```
{
  "CustomKeyId": cks-1234567890abcdef0
}
```

オペレーションが失敗した場合は、例外で示されているエラーを修正して、もう一度試してください。その他のヘルプについては、「[外部キーストアのトラブルシューティング](#)」を参照してください。

次の手順: 外部キーストアを使用するには、[これを外部キーストアプロキシに接続します](#)。

外部キーストアプロパティを編集する

既存の外部キーストアの選択したプロパティは、編集が可能です。

一部のプロパティは、外部キーストアが接続または切断されている間、編集することができます。それ以外のプロパティでは、先に、外部キーストアプロキシから[外部キーストアを切断する](#)必要があります。外部キーストアの[接続ステータス](#)は DISCONNECTED でなければなりません。外部キーストアが切断されている間はキーストアとその KMS キーを管理することはできませんが、外部キーストアで KMS キーを作成または使用することはできません。外部キーストアの[接続ステータス](#)を見つけるには、[DescribeCustomKeyStores](#) オペレーションを使用するか、外部キーストアの詳細ページで [General configuration] (一般設定) を表示します。

外部キーストアのプロパティを更新する前に、は新しい値を使用して [GetHealthStatus](#) リクエストを外部キーストアプロキシ AWS KMS に送信します。リクエストが成功すると、更新されたプロパティ値を使って外部キーストアプロキシに接続し、これを認証できることが示されます。リクエストが失敗すると、編集オペレーションは失敗し、エラーを特定する例外が生じます。

編集オペレーションが完了すると、外部キーストアの更新されたプロパティ値が AWS KMS コンソールと DescribeCustomKeyStores レスポンスに表示されます。ただし、変更が完全に有効になるまでに最大で 5 分かかります。

AWS KMS コンソールで外部キーストアを編集する場合は、プロキシ [URI パスとプロキシ認証の認証情報](#)を指定する JSON ベースの[プロキシ設定ファイル](#)をアップロードできます。一部の外部キーストアプロキシでは、このファイルは自動的に生成されます。詳細については、外部キーストアプロキシか外部キーマネージャーのドキュメントを参照してください。

 Warning

更新されたプロパティ値により、外部キーストアは、以前の値と同じ外部キーマネージャーのプロキシに接続するか、同じ暗号化キーを持つ外部キーマネージャーのバックアップまたはスナップショットのプロキシに接続します。外部キーストアが KMS キーに関連付けられた外部キーへのアクセスを完全に失うと、それらの外部キーで暗号化された暗号文は回復不能になります。特に、外部キーストアのプロキシ接続を変更すると、AWS KMS が外部キーにアクセスできなくなる可能性があります。

 Tip

外部キーマネージャーの中には、外部キーストアプロパティを編集するための簡単な方法が用意されているものもあります。詳細については外部キーマネージャーのドキュメントを参照してください。

外部キーストアの次のプロパティは、変更が可能です。

編集可能な外部キーストアプロパティ	どの接続ステータスでも可	切断済み状態が必要	
カスタムキーストア名 カスタムキーストアの必須フレンドリ名			
 Important このフィールドには、機密情報や重要情報を含めないでください。このフィールドは、CloudTrail ログやその他の出力にプレーンテキストで表示される場合があります。			
プロキシ認証の認証情報 (XksProxyAuthenticationCredential)			

編集可能な外部キーストアプロパティ	どの接続ステータスでも可	切断済み状態が必要
(1 つの要素のみを変更する場合でも、アクセスキー ID とシークレットアクセスキーの両方を指定する必要がある)。		
プロキシ URI パス (XksProxyUriPath)	✓	
プロキシ接続 (XksProxyConnectivity) (プロキシ URI エンドポイントも更新する必要がある。VPC エンドポイントサービス接続に変更する場合、プロキシ VPC エンドポイントサービス名を指定する)。		✓
プロキシ URI エンドポイント (XksProxyUriEndpoint) プロキシエンドポイント URI を変更する場合、関連付けられた TLS 証明書の変更も必要になる場合がある。		✓
プロキシ VPC エンドポイントのサービス名 (XksProxyVpcEndpointServiceName) (このフィールドは VPC エンドポイントサービスの接続に必要)。		✓

外部キーストアのプロパティを編集する

外部キーストアのプロパティは、AWS KMS コンソールで編集するか、[UpdateCustomKeyStore](#) オペレーションを使用して編集できます。

AWS KMS コンソールの使用

キーストアを編集すると、編集可能な任意の値を変更できます。一部の変更では、外部キーストアを外部キーストアプロキシから切断することが必要になります。

プロキシ URI パスまたはプロキシ認証の認証情報を編集する場合は、新しい値を入力するか、新しい値を含む、外部キーストアの[プロキシ設定ファイル](#)をアップロードします。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[Custom key stores] (カスタムキーストア)、[External key stores] (外部キーストア) の順に選択します。
4. 編集する外部キーストアの行を選択します。
5. 必要に応じて、外部キーストアを外部キーストアプロキシから切断します。[Key store actions] (キーストアアクション) メニューから [Disconnect] (切断) を選択します。
6. [Key store actions] (キーストアアクション) メニューから [Edit] (編集) を選択します。
7. 1 つまたは複数の編集可能な外部キーストアプロパティを変更します。また、プロキシ URI パスとプロキシ認証の認証情報の値を含む、外部キーストア[プロキシ設定ファイル](#)をアップロードすることもできます。プロキシ設定ファイルは、ファイル内の指定された値が変更されていなくても、使用できます。
8. [Update external key store] (外部キーストアの更新) を選択します。
9. 警告を確認し、続行する場合は警告を確定し、[Update external key store] (外部キーストアの更新) を選択します。

手順が正常に完了すると、編集したプロパティについて説明するメッセージが表示されます。正常に行われなかった場合は、問題を説明し、修正方法を示すエラーメッセージが表示されます。

10. 必要に応じて、外部キーストアを再接続します。[Key store actions] (キーストアアクション) メニューから [Connect] (接続) を選択します。

外部キーストアは切断された状態にしておくことができます。ただし、切断されている間は外部キーストアで KMS キーを作成したり、[暗号化オペレーション](#)で外部キーストア内の KMS キーを使用したりすることはできません。

AWS KMS API の使用

外部キーストアのプロパティを変更するには、[UpdateCustomKeyStore](#) オペレーションを使用します。同じオペレーションで外部キーストアの複数のプロパティを変更できます。オペレーションが成功すると、は HTTP 200 レスポンスとプロパティのない JSON オブジェクト AWS KMS を返します。

外部キーストアを識別するには CustomKeyStoreId パラメータを使用します。プロパティを変更するには他のパラメータを使用します。UpdateCustomKeyStore オペレーションでは、[プロキシ設定ファイル](#)を使用できません。プロキシ設定ファイルは、AWS KMS コンソールでのみサポートされています。ただし、プロキシ設定ファイルは、外部キーストアプロキシの正しいパラメータ値を決める際に役立ちます。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

始める前に、[必要に応じて](#)、外部キーストアプロキシから[外部キーストア](#)を切断します。更新後、必要に応じて、外部キーストアプロキシに[外部キーストアを再接続](#)できます。外部キーストアは切断された状態にしておくことができますが、キーストアに新しい KMS キーを作成したり、暗号化オペレーションのためにキーストアで既存の KMS キーを使用したりするときは、先に接続しておく必要があります。

Note

AWS CLI バージョン 1.0 を使用する場合は、パラメータなどの HTTP または HTTPS 値を持つ XksProxyUriEndpoint パラメータを指定する前に、次のコマンドを実行します。

```
aws configure set cli_follow_urlparam false
```

それ以外の場合、AWS CLI バージョン 1.0 はパラメータ値をその URI アドレスにあるコンテンツに置き換え、次のエラーが発生します。

```
Error parsing parameter '--xks-proxy-uri-endpoint': Unable to retrieve
https:// : received non 200 status code of 404
```

外部キーストアの名前を変更する

最初の例では、[UpdateCustomKeyStore](#) オペレーションを使用して、カスタムキーストアのフレンドリ名を XksKeyStore に変更します。このコマンドでは、CustomKeyStoreId パラメータを使用してカスタムキーストアを識別し、CustomKeyStoreName でカスタムキーストアの新しい名前を指定します。例にある値は、すべて外部キーストアの実際の値に置き換えます。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 --new-
custom-key-store-name XksKeyStore
```

プロキシ認証の認証情報を変更する

次の例では、外部キーストアプロキシへの認証 AWS KMS に使用するプロキシ認証情報を更新します。認証情報を更新するとき、それがプロキシ上でローテーションされる場合は、このようなコマンドを使用できます。

まず、外部キーストアプロキシで認証情報を更新します。次に、この機能を使用して AWS KMS に変更を報告します。(プロキシは古い認証情報と新しい認証情報の両方を一時的にサポートするため、認証情報を更新する時間があります) AWS KMS。

変更する値が 1 つのみの場合でも、アクセスキー ID とシークレットアクセスキーの両方を認証情報で指定する必要があります。

最初の 2 つのコマンドは、認証情報値を保持する変数を設定します。UpdateCustomKeyStore オペレーションは CustomKeyStoreId パラメータを使って外部キーストアを識別します。XksProxyAuthenticationCredential パラメータをその AccessKeyId と RawSecretAccessKey フィールドで使用し、新しい認証情報を指定します。例にある値は、すべて外部キーストアの実際の値に置き換えます。

```
$ accessKeyId=access key id
$ secretAccessKey=secret access key

$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
  --xks-proxy-authentication-credential \
    AccessKeyId=$accessKeyId,RawSecretAccessKey=$secretAccessKey
```

プロキシ URI パスを変更する

次の例では、プロキシ URI パス (XksProxyUriPath) を更新します。プロキシ URI エンドポイントとプロキシ URI パスを組み合わせた値は、AWS アカウント およびリージョン内で一意でなければなりません。例にある値は、すべて外部キーストアの実際の値に置き換えます。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
  --xks-proxy-uri-path /kms/xks/v1
```

VPC エンドポイントサービス接続を変更する

次の例では、[UpdateCustomKeyStore](#) オペレーションを使用して、外部キーストアのプロキシ接続タイプを VPC_ENDPOINT_SERVICE に変更します。この変更を行うには、VPC エンドポイントサービスの接続に必要な値 (VPC エンドポイントサービス名

(XksProxyVpcEndpointServiceName)、VPC エンドポイントサービスのプライベート DNS 名を含むプロキシ URI エンドポイント (XksProxyUriEndpoint) 値など) を指定する必要があります。例にある値は、すべて外部キーストアの実際の値に置き換えます。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
    --xks-proxy-connectivity "VPC_ENDPOINT_SERVICE" \
    --xks-proxy-uri-endpoint https://myproxy-private.xks.example.com \
    --xks-proxy-vpc-endpoint-service-name com.amazonaws.vpce.us-east-1.vpce-svc-example
```

パブリックエンドポイント接続の変更

次の例では、外部キーストアプロキシの接続タイプを PUBLIC_ENDPOINT に変更します。この変更を行うときは、プロキシ URI エンドポイント (XksProxyUriEndpoint) の値を更新する必要があります。例にある値は、すべて外部キーストアの実際の値に置き換えます。

Note

VPC エンドポイント接続は、パブリックエンドポイント接続に比べてセキュリティが優れています。パブリックエンドポイント接続に変更するときは、外部キーストアプロキシのオンプレミスでの配置や、通信のみに VPC を使用するといった他の選択肢を先に検討します。

```
$ aws kms update-custom-key-store --custom-key-store-id cks-1234567890abcdef0 \
    --xks-proxy-connectivity "PUBLIC_ENDPOINT" \
    --xks-proxy-uri-endpoint https://myproxy.xks.example.com
```

外部キーストアを表示する

各アカウントとリージョンの外部キーストアを表示するには、AWS KMS コンソールを使用するか、[DescribeCustomKeyStores](#) オペレーションを使用します。

外部キーストアを表示すると、以下を確認できます。

- フレンドリ名、ID、キーストアタイプ、作成日など、キーストアに関する基本情報。
- [接続タイプ](#)、[プロキシ URI エンドポイント](#)と[パス](#)、現在の[プロキシ認証情報](#)の[アクセスキー ID](#)など、[外部キーストアプロキシ](#)の設定情報。
- 外部キーストアプロキシが [VPC エンドポイントサービス接続](#)を使用する場合、コンソールには VPC エンドポイントサービス名が表示されます。

- 現在の[接続状態](#)。

Note

[Disconnected] (切断) の接続状態は、外部キーストアが一度も接続されたことがないこと、または意図的に外部キーストアプロキシから切断されたことを示します。ただし、接続されている外部キーストアで KMS キーの使用が失敗する場合は、外部キーストアまたはそのプロキシに問題がある可能性があります。ヘルプについては、「[外部キーストア接続エラー](#)」を参照してください。

- 外部キーストアに関する問題の検出と解決に役立つよう設計された、[Amazon CloudWatch メトリクス](#)のグラフを含む[モニタリング](#)セクション。グラフの解釈、計画やトラブルシューティングでの使用、グラフのメトリクスに基づく CloudWatch アラームの作成については、「[外部キーストアをモニタリングする](#)」を参照してください。

外部キーストアのプロパティ

外部キーストアの次のプロパティは、AWS KMS コンソールと [DescribeCustomKeyStores](#) レスポンスに表示されます。

カスタムキーストアのプロパティ

次の値は、各カスタムキーストアの詳細ページの全般設定セクションに表示されます。これらのプロパティは、キーストアや外部キーストアを含むすべてのカスタム AWS CloudHSM キーストアに適用されます。

カスタムキーストア ID

がカスタムキーストア AWS KMS に割り当ててる一意の ID。

カスタムキーストア名

カスタムキーストア作成時にカスタムキーストアに割り当ててるフレンドリ名です。この値はいつでも変更できます。

カスタムキーストアのタイプ

カスタムキーストアのタイプです。有効な値は AWS CloudHSM (AWS_CLOUDHSM) または外部キーストア (EXTERNAL_KEY_STORE) です。カスタムキーストア作成後、タイプを変更することはできません。

作成日

カスタムキーストアが作成された日付です。この日付は、AWS リージョンの現地時間で表示されます。

接続状態

カスタムキーストアがバックアップキーストアに接続されているかどうかを示します。カスタムキーストアがバックアップキーストアに一度も接続されていないか、意図的に切断されていない限り、接続状態は DISCONNECTED です。詳細については、「[the section called “接続状態”](#)」を参照してください。

外部キーストア設定プロパティ

以下の値は、各外部キーストアの詳細ページの [External key store proxy configuration] (外部キーストアプロキシ設定) セクションと [DescribeCustomKeyStores](#) レスポンスの XksProxyConfiguration 要素に表示されます。一意性要件や、各フィールドの正しい値を決定する際のヘルプなど、各フィールドの詳細な説明については、「外部キーストアの作成」トピックの「[the section called “前提条件を構成する”](#)」を参照してください。

プロキシ接続

外部キーストアが[パブリックエンドポイント接続](#)を使用しているか、[VPC エンドポイントサービス接続](#)を使用しているかを示します。

プロキシ URI エンドポイント

が[外部キーストアプロキシ](#)への接続 AWS KMS に使用するエンドポイント。

プロキシ URI パス

がプロキシ API リクエスト AWS KMS を送信するプロキシ URI エンドポイントからのパス。
[???](#)

プロキシ認証情報: アクセスキー ID

外部キーストアプロキシに設定する[プロキシ認証情報](#)の一部です。アクセスキー ID は、認証情報のシークレットアクセスキーを識別します。

AWS KMS は SigV4 署名プロセスとプロキシ認証情報を使用して、外部キーストアプロキシへのリクエストに署名します。署名の認証情報により、外部キーストアプロキシはユーザーに代わってリクエストを認証できます AWS KMS。

VPC エンドポイントサービス名

外部キーストアをサポートする Amazon VPC エンドポイントサービスの名前です。この値は、外部キーストアが [VPC エンドポイントサービス接続](#) を使用している場合にのみ表示されます。外部キーストアプロキシを VPC 内に配置するか、VPC エンドポイントサービスを使用して、外部キーストアプロキシと安全に通信できます。

外部キーストアのプロパティを表示する

外部キーストアとそれに関連するプロパティは、AWS KMS コンソールで、または [DescribeCustomKeyStores](#) オペレーションを使用して表示できます。

AWS KMS コンソールの使用

任意のアカウントとリージョンで外部キーストアを表示するには、以下の手順に従います。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[Custom key stores] (カスタムキーストア)、[External key stores] (外部キーストア) の順に選択します。
4. 外部キーストアの詳細を表示するには、キーストア名を選択します。

AWS KMS API の使用

外部キーストアを表示するには、[DescribeCustomKeyStores](#) オペレーションを使用します。デフォルトでは、このオペレーションは、アカウントとリージョンのすべてのカスタムキーストアを返します。ただし、CustomKeyId または CustomKeyName パラメータのどちらかを使用して (両方は使用できません) 出力を特定のカスタムキーストアに制限できます。

カスタムキーストアでは、出力は、カスタムキーストア ID、名前、タイプ、およびキーストアの [接続状態](#) で構成されています。接続状態が FAILED の場合、出力にはエラーの理由を説明する ConnectionErrorCode も含まれています。外部キーストアの ConnectionErrorCode を解釈する方法については、「[外部キーストアの接続エラーコード](#)」を参照してください。

外部キーストアでは、出力に XksProxyConfiguration 要素も含まれています。この要素には、[接続タイプ](#)、[プロキシ URI エンドポイント](#)、[プロキシ URI パス](#)、および [プロキシ認証情報](#) のアクセスキー ID が含まれています。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

たとえば、次のコマンドは、アカウントとリージョンのすべてのカスタムキーストアを返します。Limit パラメータと Marker パラメータを使用して、出力のカスタムキーストアをページ分割できます。

```
$ aws kms describe-custom-key-stores
```

次のコマンドは、CustomKeyStoreName パラメータを使用して、ExampleXksPublic というフレンドリ名のサンプル外部キーストアのみを取得します。このサンプルキーストアは、パブリックエンドポイント接続を使用しています。また、外部キーストアプロキシに接続されています。

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksPublic
{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-1234567890abcdef0",
      "CustomKeyStoreName": "ExampleXksPublic",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-14T20:17:36.419000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE12345670EXAMPLE",
        "Connectivity": "PUBLIC_ENDPOINT",
        "UriEndpoint": "https://xks.example.com:6443",
        "UriPath": "/example/prefix/kms/xks/v1"
      }
    }
  ]
}
```

次のコマンドは、VPC エンドポイントサービス接続を備えたサンプル外部キーストアを取得します。この例では、外部キーストアがその外部キーストアプロキシに接続されています。

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-9876543210fedcba9",
      "CustomKeyStoreName": "ExampleXksVpc",

```

```

    "ConnectionState": "CONNECTED",
    "CreationDate": "2022-12-13T18:34:10.675000+00:00",
    "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
    "XksProxyConfiguration": {
      "AccessKeyId": "ABCDE98765432EXAMPLE",
      "Connectivity": "VPC_ENDPOINT_SERVICE",
      "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
      "UriPath": "/example/prefix/kms/xks/v1",
      "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
    }
  }
]
}

```

Disconnected の [ConnectionState](#) は、外部キーストアが一度も接続されたことがないこと、または意図的に外部キーストアプロキシから切断されたことを示します。ただし、接続されている外部キーストアで KMS キーの使用が失敗する場合は、外部キーストアプロキシまたは他の外部コンポーネントに問題がある可能性があります。

外部キーストアの `ConnectionState` が `FAILED` の場合、`DescribeCustomKeyStores` レスポンスには、エラーの理由を説明する `ConnectionErrorCode` 要素が含まれています。

例えば、次の出力では、`XKS_PROXY_TIMED_OUT`値は が外部キーストアプロキシに接続 AWS KMS できることを示しますが、外部キーストアプロキシが割り当てられた時間に に応答しなかったため、接続 AWS KMS が失敗しました。この接続エラーコードが繰り返し表示される場合は、外部キーストアプロキシベンダーに通知してください。このエラーやその他の接続エラーに関するヘルプについては、「」を参照してください [外部キーストアのトラブルシューティング](#)。

```

$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-9876543210fedcba9",
      "CustomKeyName": "ExampleXksVpc",
      "ConnectionState": "FAILED",
      "ConnectionErrorCode": "XKS_PROXY_TIMED_OUT",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyStoreType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",

```

```
"UriPath": "/example/prefix/kms/xks/v1",
"VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
}
}
]
}
```

外部キーストアをモニタリングする

AWS KMS は、外部キーストアとのやり取りごとにメトリクスを収集し、CloudWatch アカウントに公開します。これらのメトリクスは、各外部キーストアの詳細ページのモニタリングセクションで、グラフを生成するために使用されます。次のトピックでは、グラフを使用して、外部キーストアに影響する運用上および設定上の問題を特定し、トラブルシューティングする方法の詳細を説明します。CloudWatch メトリクスを使用して、外部キーストアが想定どおりに動作しない場合に通知するアラームを設定することをおすすめします。詳細については、「[Amazon CloudWatch によるモニタリング](#)」を参照してください。

トピック

- [グラフを表示する](#)
- [グラフの解釈](#)

グラフを表示する

さまざまな詳細レベルでグラフを表示できます。デフォルトでは、各グラフは 3 時間の時間範囲と 5 分の[集計期間](#)を使用します。コンソール内でグラフビューを調整できますが、外部キーストアの詳細ページを閉じるかブラウザを更新すると、変更はデフォルト設定に戻ります。Amazon CloudWatch の用語のヘルプについては、「[Amazon CloudWatch の概念](#)」を参照してください。

データポイントの詳細を表示する

各グラフのデータは、[AWS KMS メトリクス](#)によって収集されます。特定のデータポイントに関する詳細を表示するには、折れ線グラフ上のデータポイントにマウスカーソルを合わせます。これにより、グラフの生成元であるメトリクスに関する詳細情報のポップアップが表示されます。各リスト項目には、そのデータポイントで記録された[ディメンション](#)値が表示されます。そのデータポイントのディメンション値に使用できるメトリクスデータがない場合、ポップアップには NULL 値 (—) が表示されます。一部のグラフでは、1 つのデータポイントに対して複数のディメンションと値が記録されます。[信頼性グラフ](#)などの他のグラフでは、メトリクスによって収集されたデータを使用して固有の値が計算されます。各リスト項目は、各折れ線グラフの色に関連付けられています。

時間範囲を変更する

グラフの[時間範囲](#)を変更するには、モニタリングセクションの右上隅にある事前定義済みの時間範囲の 1 つを選択します。事前定義済みの時間範囲は、1 時間から 1 週間です (1 時間、3 時間、12 時間、1 日、3 日、1 週間)。これにより、すべてのグラフの時間範囲が調整されます。特定のグラフを別の時間範囲で表示したい場合、またはカスタムの時間範囲を設定したい場合は、グラフを拡大するか、Amazon CloudWatch コンソールに表示します。

グラフを拡大する

[ミニマップズーム機能](#)を使用すると、ズームインビューとズームアウトビュー間を変更することなく、折れ線グラフと積み上げ面グラフのセクションに焦点を合わせることができます。例えば、ミニマップズーム機能を使用して折れ線グラフのピークに焦点を合わせると、同じタイムラインのモニタリングセクション内の他のグラフに対してスパイクを比較できます。

1. 焦点を合わせるグラフの領域を選択してドラッグし、マウスボタンを放します。
2. ズームをリセットするには、[Reset zoom] (ズームのリセット) アイコンを選択します。これは、内側にマイナス (-) 記号が付いた虫眼鏡のような見た目です。

グラフを拡大する

グラフを拡大するには、個々のグラフの右上隅にあるメニューアイコンを選択してから、[Enlarge] (拡大) を選択します。グラフにカーソルを合わせるとメニューアイコンの横に表示される、拡大アイコンを選択することもできます。

グラフを拡大すると、別の期間、カスタム時間範囲、更新間隔を指定して、グラフの表示をさらに変更できます。これらの変更は、拡大表示を閉じるとデフォルト設定に戻ります。

期間を変更する

1. [Period options] (期間オプション) メニューを選択します。デフォルトでは、このメニューには [5 minutes] (5 分) の値が表示されます。
2. 期間を選択します。事前定義された期間は 1 秒から 30 日です。

たとえば、1 分間の表示を選択できます。これは、トラブルシューティング時に役立ちます。または、詳細度がより低い 1 時間表示を選択します。これは、時間の経過に伴う傾向を確認できるように、より広い期間 (3 日間など) を表示するときに便利です。詳細については、『Amazon CloudWatch ユーザーガイド』の「[期間](#)」を参照してください。

時間範囲またはタイムゾーンを変更する

1. 1 時間 ~ 1 週間の事前定義済み時間範囲 (1 時間、3 時間、12 時間、1 日、3 日、1 週間) から 1 つを選択します。また、[Custom] (カスタム) を選択して独自の時間範囲を設定することもできます。
2. [Custom] (カスタム) を選択します。
 - a. 時間範囲ボックスの左上隅にある [Absolute] (絶対値) タブを選択します。カレンダーのピッカーまたはテキストフィールドボックスを使用して、時間範囲を指定します。
 - b. タイムゾーン: ボックスの右上隅にあるドロップダウンを選択します。タイムゾーンは [UTC] または [Local time zone] (ローカルタイムゾーン) に変更できます。
3. 時間範囲を指定したら、[Apply] (適用) を選択します。

グラフのデータ更新頻度を変更する

1. 右上隅にある [Refresh options] (更新オプション) メニューを選択します。
2. 更新間隔 (オフ、10 秒、1 分、2 分、5 分、15 分) を選択します。

Amazon CloudWatch コンソールでグラフを表示する

モニタリングセクションのグラフは、AWS KMS が Amazon CloudWatch に公開する事前定義済みのメトリクスから生成されます。グラフを CloudWatch コンソール内で開き、CloudWatch ダッシュボードに保存できます。外部キーストアが複数ある場合は、CloudWatch でそれぞれのグラフを開いて 1 つのダッシュボードに保存し、その状態と使用状況を比較できます。

Amazon CloudWatch ダッシュボードに追加する

すべてのグラフを Amazon CloudWatch ダッシュボードに追加するには、右上隅にある [Add to dashboard] (ダッシュボードに追加) を選択します。既存のダッシュボードを選択するか、新しいロールを作成できます。このダッシュボードを使用して、グラフとアラームのカスタマイズされたビューを作成する方法については、『Amazon CloudWatch ユーザーガイド』の「[Amazon CloudWatch ダッシュボードの使用](#)」を参照してください。

CloudWatch メトリクスで表示する

個々のグラフの右上隅にあるメニューアイコンを選択し、[View in metrics] (メトリクスで表示) を選択すると、このグラフが Amazon CloudWatch コンソールに表示されます。CloudWatch コンソールからこの 1 つのグラフをダッシュボードに追加して、時間範囲、期間、更新間隔を変更できます。

詳細については、『Amazon CloudWatch ユーザーガイド』の「[メトリクスのグラフ化](#)」を参照してください。

グラフの解釈

AWS KMS には、AWS KMS コンソール内の外部キーストアの状態をモニタリングするためのグラフがいくつか用意されています。これらのグラフは自動的に設定され、[AWS KMS メトリクス](#)から生成されます。

グラフデータは、外部キーストアと外部キーへの呼び出しの一部として収集されます。呼び出しを行わなかった時間範囲内にグラフにデータが入力される場合があります。このデータは、外部キーストアプロキシと外部キーマネージャーのステータスを確認するために、ユーザーに代わって AWS KMS 行う定期的な GetHealthStatus 呼び出しから取得されます。グラフに [No data available] (利用可能なデータなし) というメッセージが表示された場合は、その時間帯に呼び出しが記録されなかったか、外部キーストアが [DISCONNECTED](#) 状態であったことを示します。[ビューをより広い時間範囲に調整する](#)ことで、外部キーストアが切断された時間を特定できる場合があります。

トピック

- [Total requests](#)
- [信頼性](#)
- [レイテンシー](#)
- [例外の上位 5 位](#)
- [証明書の有効期限日数](#)

Total requests

特定の時間範囲内に特定の外部キーストアに対して受信された AWS KMS リクエストの合計数。このグラフを使用して、スロットリングのリスクがあるかどうかを判断します。

AWS KMS では、外部キーマネージャーが 1 秒あたり最大 1,800 件の暗号化オペレーションのリクエストを処理できることをお勧めします。5 分間で呼び出しが 54 万件近くになると、スロットリングのリスクが生じます。

[ExternalKeyStoreThrottle](#) メトリクスで AWS KMS スロットリングする外部キーストアの KMS キーに対する暗号化オペレーションのリクエスト数をモニタリングできます。

「リクエスト率が極めて高いため」、リクエストが拒否されたことを説明するメッセージが表示され、KMSInvalidStateException エラーが頻発する場合は、外部キーマネージャーまたは外部

キーストアプロキシが現在のリクエストレートに対応できていない可能性があります。可能な場合は、リクエスト率を下げます。また、カスタムキーストアのリクエストクォータ値の引き下げをリクエストすることも検討してください。このクォータ値を減らすとスロットリングが増加する可能性があります。AWS KMS は余分なリクエストを外部キーストアプロキシまたは外部キーマネージャーに送信される前にすぐに拒否していることを示します。クォータの削減をリクエストするには、[AWS サポート センター](#)にアクセスしてケースを作成してください。

リクエストの合計のグラフは、外部キーストアプロキシから AWS KMS が受信した成功と失敗の両方のレスポンスに関するデータを収集する、[XksProxyErrors](#) メトリクスから生成されます。[特定のデータポイントを表示する](#)と、ポップアップに、そのデータポイントで記録された AWS KMS リクエストの合計数とともに CustomKeyStoreId デイメンションの値が表示されます。CustomKeyStoreId は常に同じになります。

信頼性

外部キーストアプロキシが成功したレスポンスまたは再試行不可能なエラーを返した AWS KMS リクエストの割合。このグラフを使用して、外部キーストアプロキシのオペレーション状態を評価します。

グラフに 100% 未満の値が表示されている場合は、プロキシが応答しなかったか、再試行可能なエラーで応答したことを示します。これは、ネットワークの問題、外部キーストアプロキシまたは外部キーマネージャーの速度低下、または実装上のバグを示している可能性があります。

リクエストに不正な認証情報が含まれていてプロキシが AuthenticationFailedException で応答した場合でも、プロキシは[外部キーストアプロキシ API リクエスト](#)で誤った値を識別するため、グラフには 100% の信頼性が表示され、失敗することが予想されます。信頼性グラフのパーセンテージが 100% の場合、外部キーストアプロキシは想定どおりに応答しています。グラフに 100% 未満の値が表示されている場合、プロキシは再試行可能なエラーで応答したか、タイムアウトしています。例えば、リクエスト率が極めて高いためにプロキシが「ThrottlingException」を返した場合、プロキシはリクエスト失敗の原因となった特定の問題を識別できないため、信頼性の割合が低下します。これは、再試行可能なエラーは一時的な問題である可能性が高く、リクエストを再試行することで解決できるためです。

次のエラーレスポンスは、信頼性の割合を低下させます。[例外の上位 5 位](#) グラフと [XksProxyErrors](#) メトリクスを使用して、プロキシが再試行可能な各エラーを返す頻度をさらにモニタリングすることができます。

- InternalException
- DependencyTimeoutException

- ThrottlingException
- XksProxyUnreachableException

信頼性グラフは、[XksProxyErrors](#) メトリクスから算出されます。メトリクスは、が外部キーストアプロキシから AWS KMS 受信した成功したレスポンスと失敗したレスポンスの両方に関するデータを収集します。信頼性の割合は、レスポンスが Retryable の ErrorType 値をもつ場合にのみ低下します。[特定のデータポイントを表示する](#)と、ポップアップに、そのデータポイントで記録された AWS KMS リクエストの信頼性の割合とともに CustomKeyId デイメンションの値が表示されます。CustomKeyId は常に同じになります。

[XksProxyErrors](#) メトリクスを使用して、1 分間に再試行可能なエラーが 5 回以上記録された場合に警告することで、潜在的なネットワーク上の問題を通知する CloudWatch アラームを作成することをお勧めします。詳細については、「[再試行可能なエラーのアラームを作成する](#)」を参照してください。

レイテンシー

外部キーストアプロキシが AWS KMS リクエストに応答するのにかかるミリ秒数。このグラフを使用して、外部キーストアプロキシと外部キーマネージャーのパフォーマンスを評価します。

AWS KMS は、外部キーストアプロキシが 250 ミリ秒以内に各リクエストに応答することを期待します。ネットワークタイムアウトの場合、AWS KMS はリクエストを 1 回再試行します。プロキシが 2 回失敗した場合、記録されるレイテンシーは、両方のリクエスト試行のタイムアウト制限を合わせたもので、グラフには約 500 ミリ秒が表示されます。それ以外の、プロキシが 250 ミリ秒のタイムアウト制限内に応答しないすべての場合、記録されるレイテンシーは 250 ミリ秒です。プロキシが暗号化と復号オペレーションで頻繁にタイムアウトする場合は、外部プロキシ管理者に相談してください。レイテンシー問題のトラブルシューティングについては、「[レイテンシーとタイムアウトエラー](#)」を参照してください。

応答が遅い場合は、外部キーマネージャーが現在のリクエストトラフィックを処理できないことを示している場合もあります。AWS KMS では、外部キーマネージャーが 1 秒あたり最大 1,800 件の暗号化オペレーションのリクエストを処理できることをお勧めします。外部キーマネージャーが 1 秒あたり 1,800 件のリクエストを処理できない場合は、[カスタムキーストアの KMS キーリクエストクォータ](#)の引き下げをリクエストすることを検討してください。外部キーストアの KMS キーを使用した暗号化オペレーションのリクエストは、外部キーストアプロキシまたは外部キーマネージャーによって処理され、後で拒否されるのではなく、[スロットリング](#)例外でフェイルファストします。

レイテンシーグラフは [XksProxyLatency](#) メトリクスから生成されます。[特定のデータポイントを表示する](#)と、ポップアップには KmsOperation および XksOperation のデイメンション値とともに

に、そのデータポイントで記録されたオペレーションの平均レイテンシーが表示されます。リスト項目は、最大レイテンシーから最低レイテンシーの順に並べられます。

[XksProxyLatency](#) メトリクスを使用して、レイテンシーがタイムアウト制限に近づいたときに通知する CloudWatch アラームを作成することをお勧めします。詳細については、「[レスポンスタイムアウトのアラームを作成する](#)」を参照してください。

例外の上位 5 位

任意の時間範囲で暗号化オペレーションと管理オペレーションが失敗した場合の、例外の上位 5 位です。このグラフを使用して、最も頻発するエラーを追跡することで、エンジニアリング作業に優先順位を付けることができます。

この数には、 が外部キーストアプロキシから AWS KMS 受け取った例外と XksProxyUnreachableException、外部キーストアプロキシとの通信を確立できない場合に内部で AWS KMS 返される例外が含まれます。

再試行可能なエラーの発生率が高い場合はネットワークエラーの可能性、再試行できないエラーの発生率が高い場合は、外部キーストアの設定に関する問題である可能性があります。例えば、のスパイクは、 で設定された認証情報 AWS KMS と外部キーストアプロキシの間に不一致がある AuthenticationFailedExceptions ことを示します。外部キーストア設定を確認するには、「[外部キーストアを表示する](#)」を参照してください。外部キーストア設定を編集するには、「[外部キーストアプロパティを編集する](#)」を参照してください。

が外部キーストアプロキシから AWS KMS 受け取る例外は、オペレーションが失敗した場合に が AWS KMS 返す例外とは異なります。AWS KMS 暗号化オペレーションは、外部キーストアの外部設定または接続状態に関連するすべての障害 KMSInvalidStateException に対して を返します。問題を特定するには、添付のエラーメッセージテキストを使用します。

次の表は、上位 5 つの例外グラフに表示される可能性のある例外と、 から AWS KMS 返される対応する例外を示しています。

エラータイプ	グラフに表示される例外	から AWS KMS 返された例外
再試行不可	AccessDeniedException トラブルシューティングのヘルプについては、「 プロキシ	CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException 。

エラータイプ	グラフに表示される例外 の承認に関する問題」を参照してください。	から AWS KMS 返された例外 暗号化オペレーションに対応する KMSInvalidStateException。
再試行不可	AuthenticationFailedException トラブルシューティングのヘルプについては、「認証情報エラー」を参照してください。	CreateCustomKeyStore および UpdateCustomKeyStore オペレーションに対応する XksProxyIncorrectAuthenticationCredentialException CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException。 暗号化オペレーションに対応する KMSInvalidStateException。
再試行可能	DependencyTimeoutException トラブルシューティングのヘルプについては、「レイテンシーとタイムアウトエラー」を参照してください。	CreateCustomKeyStore および UpdateCustomKeyStore オペレーションに対応する XksProxyUriUnreachableException CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException。 暗号化オペレーションに対応する KMSInvalidStateException。

エラータイプ	グラフに表示される例外	から AWS KMS 返された例外
再試行可能	InternalException 外部キーマネージャーと通信できないため、外部キーストアプロキシがリクエストを拒否しました。外部キーストアプロキシ設定が正しく、外部キーマネージャーが使用可能であることを検証します。	CreateCustomKeyStore および UpdateCustomKeyStore オペレーションに対応する XksProxyInvalidResponseException CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException 。 暗号化オペレーションに対応する KMSInvalidStateException 。
再試行不可	InvalidCiphertextException トラブルシューティングのヘルプについては、「 復号エラー 」を参照してください。	暗号化オペレーションに対応する KMSInvalidStateException 。
再試行不可	InvalidKeyUsageException トラブルシューティングのヘルプについては、「 外部キーの暗号化オペレーションエラー 」を参照してください。	CreateKey オペレーションに対応する XksKeyInvalidConfigurationException 。 暗号化オペレーションに対応する KMSInvalidStateException 。

エラータイプ	グラフに表示される例外	から AWS KMS 返された例外
再試行不可	InvalidStateException トラブルシューティングのヘルプについては、「 外部キーの暗号化オペレーションエラー 」を参照してください。	CreateKey オペレーションに対応する XksKeyInvalidConfigurationException 。 暗号化オペレーションに対応する KMSInvalidStateException 。
再試行不可	InvalidUriPathException トラブルシューティングのヘルプについては、「 一般的な設定エラー 」を参照してください。	CreateCustomKeyStore および UpdateCustomKeyStore オペレーションに対応する XksProxyInvalidConfigurationException CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException 。 暗号化オペレーションに対応する KMSInvalidStateException 。
再試行不可	KeyNotFoundException トラブルシューティングのヘルプについては、「 外部キーエラー 」を参照してください。	CreateKey オペレーションに対応する XksKeyNotFoundException 。 暗号化オペレーションに対応する KMSInvalidStateException 。

エラータイプ	グラフに表示される例外	から AWS KMS 返された例外
再試行可能	ThrottlingException リクエスト率が極めて高いため、外部キーストアプロキシがリクエストを拒否しました。この外部キーストアの KMS キーを使用して、呼び出しの頻度を減らします。	CreateCustomKeyStore および UpdateCustomKeyStore オペレーションに対応する XksProxyUriUnreachableException CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException 。 暗号化オペレーションに対応する KMSInvalidStateException 。
再試行不可	UnsupportedOperationException トラブルシューティングのヘルプについては、「 外部キーの暗号化オペレーションエラー 」を参照してください。	CreateKey オペレーションに対応する XksKeyInvalidResponseException 。 暗号化オペレーションに対応する KMSInvalidStateException 。

エラータイプ	グラフに表示される例外	から AWS KMS 返された例外
再試行不可	ValidationException トラブルシューティングのヘルプについては、「 プロキシの問題 」を参照してください。	CreateCustomKeyStore および UpdateCustomKeyStore オペレーションに対応する XksProxyInvalidResponseException CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException 。 暗号化オペレーションに対応する KMSInvalidStateException 。
再試行可能	XksProxyUnreachableException このエラーが繰り返し表示される場合は、外部キーストアプロキシがアクティブでネットワークに接続されていること、および外部キーストアの URI パスとエンドポイント URI または VPC サービス名が正しいことを確認します。	CreateCustomKeyStore および UpdateCustomKeyStore オペレーションに対応する XksProxyUnreachableException CreateKey オペレーションに対応する CustomKeyStoreInvalidStateException 。 暗号化オペレーションに対応する KMSInvalidStateException 。

上位 5 位の例外グラフは、[XksProxyErrors](#) メトリクスから生成されます。[任意のデータポイントを表示する](#)と、ポップアップに ExceptionName デイメンションの値と、そのデータポイントで例外

が記録された回数が表示されます。5 つのリスト項目は、最高頻度から最低頻度の例外の順に並べられます。

[XksProxyErrors](#) メトリクスを使用して、1 分間に再試行不可能なエラーが 5 回以上記録された場合に警告することで、潜在的な設定問題を通知する CloudWatch アラームを作成することをお勧めします。詳細については、「[再試行不可能なエラーのアラームを作成する](#)」を参照してください。

証明書の有効期限日数

外部キーストアプロキシエンドポイント (XksProxyUriEndpoint) の TLS 証明書の有効期限が切れるまでの日数です。このグラフを使用して、TLS 証明書の有効期限をモニタリングします。

証明書の有効期限が切れると、AWS KMS は外部キーストアプロキシと通信できません。証明書が更新されるまで、外部キーストアの KMS キーで保護されているすべてのデータにアクセスできなくなります。

証明書の有効期限までの日数のグラフは、[XksProxyCertificateDaysToExpire](#) メトリクスから生成されます。このメトリクスを使用して、有効期限切れの接近を通知する CloudWatch アラームを作成することを強くお勧めします。証明書の有効期限が切れると、暗号化されたリソースにアクセスできなくなる可能性があります。アラームを設定することで、組織は有効期限が切れる前に証明書を更新する時間をもつことができます。詳細については、「[証明書の有効期限切れアラームを作成する](#)」を参照してください。

外部キーストアを接続および切断する

新しい外部キーストアが接続されていません。外部キーストア AWS KMS keys で を作成して使用するには、外部キーストアを[外部キーストアプロキシ](#)に接続する必要があります。外部キーストアはいつでも接続と切断ができ、[その接続ステータスを表示できます](#)。

外部キーストアが切断されている間、AWS KMS は外部キーストアプロキシと通信できません。それにより、外部キーストアと既存の KMS キーの表示と管理が行えます。ただし、外部キーストアに KMS キーを作成したり、その KMS キーを暗号化オペレーションに使用したりすることはできません。プロパティを編集するときなど外部キーストアの切断が必要になる場合があります。状況に応じて適切に計画を立てます。キーストアを切断すると、その KMS キーを使用する AWS サービスのオペレーションが中断される可能性があります。

外部キーストアを自分で接続する必要はありません。外部キーストアは、無期限に切断状態のままにし、使用の必要がある場合のみ接続することができます。ただし、定期的に接続をテストして、接続が正しく接続できることを確認するとよいでしょう。

カスタムキーストアを切断すると、そのキーストアの KMS キーはただちに使用できなくなります (結果整合性の影響を受ける)。ただし、KMS キーで保護された [データキー](#) により暗号化されたリソースは、KMS キーがデータキーの復号化などで再び使用されるまでは影響を受けません。この問題は AWS のサービスに影響します。その多くが、リソースを保護するためにデータキーを使用しています。詳細については、「[使用できない KMS キーがデータキーに及ぼす影響](#)」を参照してください。

Note

外部キーストアは、一度も接続されたことがないか明示的に切断した場合のみ DISCONNECTED ステータスになります。CONNECTED ステータスは、外部キーストアやそのサポートコンポーネントが効率的に機能していることを示すものではありません。外部キーストアコンポーネントのパフォーマンスについては、各外部キーストアの詳細ページにある [Monitoring] (モニタリング) セクションのグラフを参照してください。詳細については、「[外部キーストアをモニタリングする](#)」を参照してください。

外部キーマネージャーは、外部キーストアと AWS KMS 外部キーストアプロキシ間、または外部キーストアプロキシと外部キーマネージャー間の通信を停止および再開する追加の方法を提供する場合があります。詳細については外部キーマネージャーのドキュメントを参照してください。

トピック

- [接続状態](#)
- [外部キーストアを接続する](#)
- [外部キーストアを切断する](#)

接続状態

接続と切断により、カスタムキーストアの接続ステータスが変更されます。接続状態の値は、AWS CloudHSM キーストアと外部キーストアで同じです。

カスタムキーストアの接続状態を表示するには、[DescribeCustomKeyStores](#) オペレーションまたは AWS KMS コンソールを使用します。[Connection state] (接続ステータス) は、各カスタムキーストアテーブルに表示され、各カスタムキーストアの詳細ページ内の [General configuration] (一般設定) のセクション、およびカスタムキーストアの KMS キーの [Cryptographic configuration] (暗号化設定) タブに表示されます。詳細については、「[AWS CloudHSM キーストアを表示する](#)」および「[外部キーストアを表示する](#)」を参照してください。

カスタムキーストアの接続ステータスは、次のいずれかになります。

- **CONNECTED:** カスタムキーストアはバックアップキーストアに接続された状態です。ユーザーは、カスタムキーストアで KMS キーを作成し使用することができます。

キーストアのバックアップ AWS CloudHSM キーストアは、関連付けられた AWS CloudHSM クラスターです。外部キーストアのバックアップキーストアは、外部キーストアプロキシと、それがサポートしている外部キーマネージャーです。

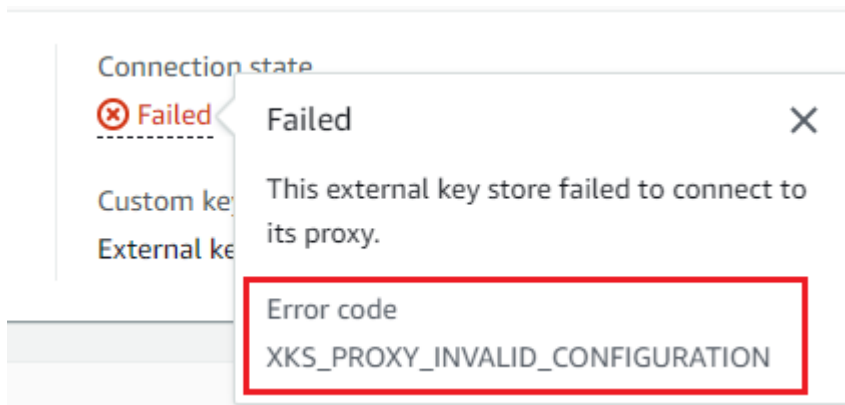
CONNECTED ステータスは、接続に成功し、カスタムキーストアが意図的に切断されていない状態を示します。接続が正常に機能していることを示すものではありません。AWS CloudHSM キーストアに関連付けられた AWS CloudHSM クラスターのステータスについては、「AWS CloudHSM ユーザーガイド」の「の [CloudWatch メトリクスの取得 AWS CloudHSM](#)」を参照してください。外部キーストアのステータスとオペレーションについては、各外部キーストアの詳細ページにある [Monitoring] (モニタリング) セクションのグラフを参照してください。詳細については、「[外部キーストアをモニタリングする](#)」を参照してください。

- **CONNECTING:** カスタムキーストアの接続処理が進行している状態です。これは過渡的な状態です。
- **DISCONNECTED:** カスタムキーストアがバックアップに接続されていないか、AWS KMS コンソールまたは [DisconnectCustomKeyStore](#) オペレーションを使用して意図的に切断されました。
- **DISCONNECTING:** カスタムキーストアを接続する処理が進行中です。これは過渡的な状態です。
- **FAILED:** カスタムキーストアを接続しようとして失敗した状態です。[DescribeCustomKeyStores](#) レスポンスの `ConnectionErrorCode` は問題を示します。

カスタムキーストアを接続するには、接続ステータスが DISCONNECTED 担っていなければなりません。接続ステータスが FAILED である場合、`ConnectionErrorCode` を使用して問題を特定し、解決します。カスタムキーストアを切断し、再度接続を試みます。接続障害については、[外部キーストア接続エラー](#) を参照してください。接続エラーコードの対処方法については「[外部キーストアの接続エラーコード](#)」を参照してください。

接続エラーコードを表示するには:

- [DescribeCustomKeyStores](#) レスポンスで、`ConnectionErrorCode` 要素の値を表示します。この要素は、`ConnectionState` が FAILED の場合にのみ、[DescribeCustomKeyStores](#) レスポンスに表示されます。
- AWS KMS コンソールで接続エラーコードを表示するには、外部キーストアの詳細ページで Failed 値にカーソルを合わせます。



外部キーストアを接続する

外部カスタムキーストアを外部キーストアプロキシに接続しているときは、[外部キーストアに KMS キーを作成](#)し、その既存の KMS キーを[暗号化オペレーション](#)に使用できます。

外部キーストアを外部キーストアプロキシに接続するプロセスは、外部キーストアの接続性に応じて異なります。

- 外部キーストアを[パブリックエンドポイント接続](#)で接続すると、AWS KMS は、[GetHealthStatus リクエスト](#)を外部キーストアプロキシに送信して、[プロキシ URI エンドポイント](#)、[プロキシ URI パス](#)、[プロキシ認証の認証情報](#)を検証します。プロキシからのレスポンスにより、[プロキシ URI エンドポイント](#)と[プロキシ URI パス](#)が正確かつアクセス可能であること、および、外部キーストアの[プロキシ認証の認証情報](#)で署名されたリクエストをプロキシが認証したことが確定します。
- 外部キーストアを[VPC エンドポイントサービス接続](#)で外部キーストアプロキシに接続すると、は以下 AWS KMS を実行します。
 - [プロキシ URI エンドポイント](#)で指定されたプライベート DNS 名のドメインが[検証済み](#)であることを確認する。
 - AWS KMS VPC から VPC エンドポイントサービスへのインターフェイスエンドポイントを作成します。
 - プロキシ URI エンドポイントで指定された、プライベート DNS 名のプライベートホストゾーンを作成する。
 - [GetHealthStatus リクエスト](#)を外部キーストアプロキシに送信する。プロキシからのレスポンスにより、[プロキシ URI エンドポイント](#)と[プロキシ URI パス](#)が正確かつアクセス可能であること、および、外部キーストアの[プロキシ認証の認証情報](#)で署名されたリクエストをプロキシが認証したことが確定します。

接続オペレーションによりカスタムキーストアを接続するプロセスが開始しますが、外部キーストアが外部プロキシに接続されるまでに約 5 分かかります。接続オペレーションからのレスポンスは、外部キーストアが接続されたことを示すものではありません。接続が成功したことを確認するには、AWS KMS コンソールまたは [DescribeCustomKeyStores](#) オペレーションを使用して、キーストアの外部[接続状態](#)を表示します。

接続状態が の場合 FAILED、コンソールに接続エラーコードが表示され AWS KMS、DescribeCustomKeyStore レスポンスに追加されます。接続エラーコードの解釈については、「[外部キーストアの接続エラーコード](#)」を参照してください。

外部キーストアに接続および再接続する

AWS KMS コンソールで、または [ConnectCustomKeyStore](#) オペレーションを使用して、外部キーストアに接続または再接続できます。

AWS KMS コンソールの使用

AWS KMS コンソールを使用して、外部キーストアを外部キーストアプロキシに接続できます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセクターを使用します。
3. ナビゲーションペインで、[Custom key stores] (カスタムキーストア)、[External key stores] (外部キーストア) の順に選択します。
4. 接続する外部キーストアの行を選択します。

外部キーストアの[接続ステータス](#)が FAILED である場合、[外部キーストアを切断](#)してから接続します。

5. [Key store actions] (キーストアアクション) メニューから [Connect] (接続) を選択します。

通常、接続プロセスが完了するまでに約 5 分かかります。オペレーションが完了すると、[接続ステータス](#)は CONNECTED に変わります。

接続状態が Failed になった場合は、接続ステータスにカーソルを合わせると、エラーの原因を示す接続エラーコードが表示されます。接続エラーコードの対処方法については「[外部キーストアの接続エラーコード](#)」を参照してください。接続ステータスが Failed である外部キーストアに接続するには、先に[カスタムキーストアを切断](#)します。

AWS KMS API の使用

切断された外部キーストアを接続するには、[ConnectCustomKeyStore](#) オペレーションを使用します。

接続する前、外部キーストアの[接続ステータス](#)は DISCONNECTED になっているはずです。接続ステータスが FAILED である場合は、[外部キーストアを切断](#)してから接続します。

この接続処理は、約 5 分で完了します。すぐに失敗しない限り、ConnectCustomKeyStore は、HTTP 200 レスポンスと、プロパティを含まない JSON オブジェクトを返します。ただし、この初期レスポンスは接続に成功したことを示していません。外部キーストアが接続されているかどうかを確認するには、[DescribeCustomKeyStores](#) のレスポンスの接続ステータスを確認します。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

外部キーストアを識別するには、カスタムキーストア ID を使用します。ID はコンソールの [カスタムキーストア] ページまたは [DescribeCustomKeyStores](#) オペレーションを使用して見つけることができます。この例を実行する前に、例の ID を有効な ID に置き換えます。

```
$ aws kms connect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

ConnectCustomKeyStore オペレーションは、そのレスポンスで ConnectionState を返しません。外部キーストアが接続されていることを確認するには、[DescribeCustomKeyStores](#) オペレーションを使用します。デフォルトでは、このオペレーションは、アカウントとリージョンのすべてのカスタムキーストアを返します。ただし、CustomKeyId または CustomKeyName パラメータのどちらかを使用して (両方は使用できません) レスポンスを特定のカスタムキーストアに制限できます。CONNECTED の ConnectionState 値は、外部キーストアが外部キーストアプロキシに接続されていることを示します。

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-9876543210fedcba9",
      "CustomKeyName": "ExampleXksVpc",
      "ConnectionState": "CONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
```



```

    "AccessKeyId": "ABCDE98765432EXAMPLE",
    "Connectivity": "VPC_ENDPOINT_SERVICE",
    "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
    "UriPath": "/example/prefix/kms/xks/v1",
    "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
  }
}
]
}

```

DescribeCustomKeyStores の ConnectionState 値が FAILED である場合、ConnectionErrorCode 要素に失敗した原因が示されます。

次の例では、の XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND 値は、が外部キーストアプロキシとの通信に使用する VPC AWS KMS エンドポイントサービスを見つけられない ConnectionErrorCode ことを示します。XksProxyVpcEndpointServiceName が正しいこと、AWS KMS サービスプリンシパルが Amazon VPC エンドポイントサービスで許可されたプリンシパルであること、および VPC エンドポイントサービスが接続リクエストを受け入れる必要がないことを確認します。接続エラーコードの対処方法については「[外部キーストアの接続エラーコード](#)」を参照してください。

```

$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-9876543210fedcba9",
      "CustomKeyName": "ExampleXksVpc",
      "ConnectionState": "FAILED",
      "ConnectionErrorCode": "XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}

```

外部キーストアを切断する

[VPC エンドポイントサービスに接続している](#) 外部キーストアを、外部キーストアプロキシから切断すると、AWS KMS は、VPC エンドポイントサービスとのインターフェイスエンドポイントを削除し、接続をサポートするために作成されたネットワークインフラストラクチャを削除します。パブリックエンドポイントに接続している外部キーストアには、同等のプロセスは必要ありません。このアクションは、VPC エンドポイントサービスやそのサポートコンポーネントには影響せず、外部キーストアプロキシや外部コンポーネントにも影響しません。

外部キーストアが切断されている間、AWS KMS は外部キーストアプロキシにリクエストを送信しません。外部キーストアの接続状態は DISCONNECTED です。切断された外部キーストアの KMS キーは、[\(削除が保留されている場合を除き\)](#) **UNAVAILABLE のキーステータス**になります。つまり、暗号化オペレーションには使用できません。しかし、外部キーストアと既存の KMS キーの表示および管理は引き続き行えます。

切断状態は一時的なものとして、また元に戻せるように設計されています。外部キーストアはいつでも再接続できます。通常、再度設定する必要はありません。ただし、関連付けられた外部キーストアプロキシのプロパティが、[プロキシ認証の認証情報のローテーション](#)などにより変更されている場合は、再度接続する前に、[外部キーストアの設定を編集する](#) 必要があります。

Note

カスタムキーストアが切断されている間は、カスタムキーストアで KMS キーを作成したり、暗号化オペレーションで既存の KMS キーを使用したりする試みはすべて失敗します。このオペレーションにより、ユーザーが機密データを保存したりアクセスしたりすることを防ぐことができます。

外部キーストアの切断による影響を、より正確に予測するには、外部キーストアで KMS キーを識別して[過去の使用状況を特定](#)します。

外部キーストアを切断する理由としては、次のようなものが挙げられます。

- プロパティを編集するには。外部キーストアが接続されている間は、カスタムキーストア名、プロキシ URI パス、プロキシ認証の認証情報の編集が行えます。ただし、プロキシの接続タイプ、プロキシ URI エンドポイント、VPC エンドポイントのサービス名を編集するには、先に外部キーストアを切断しておく必要があります。詳細については、「[外部キーストアプロパティを編集する](#)」を参照してください。

- AWS KMS と外部キーストアプロキシ間のすべての通信を停止するには。エンドポイントまたは VPC エンドポイントサービスを無効にすることで、AWS KMS とプロキシ間の通信を停止することもできます。さらに、外部キーストアプロキシまたはキー管理ソフトウェアは、AWS KMS がプロキシと通信できないようにする、またはプロキシが外部キーマネージャーにアクセスできないようにする追加のメカニズムを提供する場合があります。
- 外部キーストアですべての KMS キーを無効にするには。AWS KMS コンソールまたは [DisableKey](#) オペレーションを使用して、外部キーストアで [KMS キーを無効化および再有効化](#) できます。これらのオペレーションはすぐに完了します (結果整合性の影響を受ける) が、一度に 1 つの KMS キーしか処理されません。外部キーストアを切断すると外部キーストアのすべての KMS キーのキーステータスが Unavailable に変更され、暗号化オペレーションに使用できなくなります。
- 失敗した接続試行を修復するには。外部キーストアを接続する試みに失敗した場合 (カスタムキーストアの接続ステータスが FAILED になる) は、再度接続を試みる前に外部キーストアを切断する必要があります。

外部キーストアを切断する

AWS KMS コンソールで、または [DisconnectCustomKeyStore](#) オペレーションを使用して、外部キーストアを切断できます。

AWS KMS コンソールの使用

AWS KMS コンソールを使用して、外部キーストアを外部キーストアプロキシに接続できます。この処理は約 5 分で完了します。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクターを使用します。
3. ナビゲーションペインで、[Custom key stores] (カスタムキーストア)、[External key stores] (外部キーストア) の順に選択します。
4. 切断する外部キーストアの行を選択します。
5. [Key store actions] (キーストアアクション) メニューから [Disconnect] (切断) を選択します。

オペレーションが完了すると、接続状態が [DISCONNECTING] から [DISCONNECTED] に変わります。オペレーションが失敗した場合は、問題を説明し、修正方法を示すエラーメッセージが表示されます。さらにヘルプが必要な場合は、「[外部キーストア接続エラー](#)」を参照してください。

AWS KMS API の使用

接続している外部キーストアを切断するには、[DisconnectCustomKeyStore](#) オペレーションを使用します。オペレーションが成功すると、は HTTP 200 レスポンスとプロパティのない JSON オブジェクト AWS KMS を返します。この処理は約 5 分で完了します。外部キーストアの接続ステータスを確認するときは、[DescribeCustomKeyStores](#) オペレーションを使用します。

このセクションの例では [AWS Command Line Interface \(AWS CLI\)](#) を使用しますが、サポートされている任意のプログラミング言語を使用することができます。

こちらの例では、VPC エンドポイントサービスに接続した外部キーストアを切断します。このコマンドを実行する前に、例にあるカスタムキーストア ID を有効な ID に置き換えます。

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

外部キーストアが切断されていることを確認するには、[DescribeCustomKeyStores](#) オペレーションを使用します。デフォルトでは、このオペレーションは、アカウントとリージョンのすべてのカスタムキーストアを返します。ただし、CustomKeyId または CustomKeyName パラメータのどちらかを使用して (両方は使用できません) レスポンスを特定のカスタムキーストアに制限できます。DISCONNECTED の ConnectionState 値は、例にある外部キーストアが、外部キーストアプロキシに接続されていないことを示します。

```
$ aws kms describe-custom-key-stores --custom-key-store-name ExampleXksVpc
{
  "CustomKeyStores": [
    {
      "CustomKeyId": "cks-9876543210fedcba9",
      "CustomKeyName": "ExampleXksVpc",
      "ConnectionState": "DISCONNECTED",
      "CreationDate": "2022-12-13T18:34:10.675000+00:00",
      "CustomKeyType": "EXTERNAL_KEY_STORE",
      "XksProxyConfiguration": {
        "AccessKeyId": "ABCDE98765432EXAMPLE",
        "Connectivity": "VPC_ENDPOINT_SERVICE",
        "UriEndpoint": "https://example-proxy-uri-endpoint-vpc",
        "UriPath": "/example/prefix/kms/xks/v1",
        "VpcEndpointServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-example"
      }
    }
  ]
}
```

外部キーストアを削除する

外部キーストアを削除すると、は外部キーストアプロキシに関する情報を含め AWS KMS、外部キーストアに関するすべてのメタデータ AWS KMS を から削除します。このオペレーションは、[外部キーストアプロキシ](#)、[外部キーマネージャー](#)、[外部キー](#)、または Amazon VPC や VPC エンドポイントサービスなどの外部キーストアをサポートするために作成した AWS リソースには影響しません。

外部キーストアを削除するときは、先に、キーストアから[すべての KMS キーを削除](#)し、外部キーストアプロキシから[キーストアを切断](#)しておきます。さもないと、キーストアを削除することができません。

外部キーストアを削除すると元には戻せませんが、新しい外部キーストアを作成し、同じ外部キーストアプロキシと外部キーマネージャーに関連付けることができます。ただし、同じ外部キーマテリアルにアクセスできる場合でも、外部キーストアで対称暗号化 KMS キーを再作成することはできません。は、各 KMS キーに固有の対称暗号文にメタデータ AWS KMS を含めます。このセキュリティ機能により、データを暗号化した KMS キーのみが、そのデータを復号することができます。

外部キーストアを削除するのではなく、切断することを検討します。外部キーストアが切断されている間は、外部キーストアとその を管理できます AWS KMS keys が、外部キーストアで KMS キーを作成または使用することはできません。外部キーストアはいつでも再接続でき、KMS キーを使用して、データの暗号化と復号化を再開できます。切断された外部キーストアプロキシや使用できない KMS キーには、コストは発生しません。

AWS KMS コンソールで、または [DeleteCustomKeyStore](#) オペレーションを使用して、外部キーストアを削除できます。

AWS KMS コンソールの使用

AWS KMS コンソールを使用して、外部キーストアを削除できます。

1. にサインイン AWS Management Console し、<https://console.aws.amazon.com/kms> で AWS Key Management Service (AWS KMS) コンソールを開きます。
2. を変更するには AWS リージョン、ページの右上隅にあるリージョンセレクタを使用します。
3. ナビゲーションペインで、[Custom key stores] (カスタムキーストア)、[External key stores] (外部キーストア) の順に選択します。
4. 削除する外部キーストアが表示されている行を探します。外部キーストアの [Connection state] (接続ステータス) が DISCONNECTED になっていなければ、先に[カスタムキーストアを切断](#)し、その後に削除します。

5. [Key store actions] (キーストアアクション) メニューから [Delete] (削除) を選択します。

オペレーションが完了すると成功メッセージが表示され、この外部 キーストアはカスタムキーストアリストに表示されなくなります。オペレーションが正常に行われなかった場合、問題を説明し、修正方法を示すエラーメッセージが表示されます。さらにヘルプが必要な場合は、「[外部キーストアのトラブルシューティング](#)」を参照してください。

AWS KMS API の使用

外部キーストアを削除するときは、[DeleteCustomKeyStore](#) オペレーションを使用します。オペレーションが成功すると、は HTTP 200 レスポンスとプロパティのない JSON オブジェクト AWS KMS を返します。

まず、外部キーストアを切断します。このコマンドを実行する前に、例のカスタムキーストア ID を有効な ID に置き換えます。

```
$ aws kms disconnect-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

外部キーストアを切断すると、[DeleteCustomKeyStore](#) オペレーションでこれを削除することができます。

```
$ aws kms delete-custom-key-store --custom-key-store-id cks-1234567890abcdef0
```

外部キーストアが削除されていることを確認するには、[DescribeCustomKeyStores](#) オペレーションを使用します。

```
$ aws kms describe-custom-key-stores

{
  "CustomKeyStores": []
}
```

存在しないカスタムキーストア名または ID を指定すると、は CustomKeyStoreNotFoundException 例外 AWS KMS を返します。

```
$ aws kms describe-custom-key-stores --custom-key-store-id cks-1234567890abcdef0
```

```
An error occurred (CustomKeyStoreNotFoundException) when calling the
DescribeCustomKeyStore operation:
```

外部キーストアのトラブルシューティング

外部キーストアのほとんどの問題の解決は、各例外とともに AWS KMS 表示されるエラーメッセージ、または[外部キーストアを外部キー](#)ストアプロキシに接続しようとしたときに が AWS KMS 返す[接続エラーコード](#)によって示されます。ただし、一部の問題はもう少し複雑です。

外部キーストアの問題を診断する際は、最初に原因を特定します。これにより、対処法の範囲が狭まり、トラブルシューティングがより効率的になります。

- AWS KMS — [外部キーストア設定](#)の値が正しくないなど AWS KMS、問題が 内にある可能性があります。
- 外部 — この問題は AWS KMS、外部キーストアプロキシ、外部キーマネージャー、外部キー、または VPC エンドポイントサービスの設定またはオペレーションに関する問題など、 の外部で発生する可能性があります。
- ネットワーク — プロキシエンドポイント、ポート、IP スタック、プライベート DNS 名またはドメインの問題など、接続またはネットワークに問題がある可能性があります。

Note

外部キーストアで管理オペレーションが失敗すると、複数の異なる例外が生成されます。ただし、AWS KMS 暗号化オペレーションは、外部キーストアの外部設定または接続状態に関連するすべての障害KMSInvalidStateExceptionに対して返されます。問題を特定するには、添付のエラーメッセージテキストを使用します。

[ConnectCustomKeyStore](#) オペレーションは、接続プロセスが完了する前に、直ちに成功します。接続プロセスが成功したかどうかを判断するには、外部キーストアの[接続状態](#)を表示します。接続プロセスが失敗した場合、AWS KMS は原因を説明し、対処方法を提案する[接続エラーコード](#)を返します。

トピック

- [外部キーストアのトラブルシューティングツール](#)
- [設定エラー](#)
- [外部キーストア接続エラー](#)
- [レイテンシーとタイムアウトエラー](#)
- [認証情報エラー](#)
- [キーステータスエラー](#)

- [復号エラー](#)
- [外部キーエラー](#)
- [プロキシの問題](#)
- [プロキシの承認に関する問題](#)

外部キーストアのトラブルシューティングツール

AWS KMS には、外部キーストアとそのキーに関する問題を特定して解決するのに役立つツールがいくつか用意されています。これらのツールは、外部キーストアプロキシおよび外部キーマネージャーに付属するツールと組み合わせて使用します。

Note

外部キーストアプロキシと外部キーマネージャーを使用すると、外部キーストアとその KMS キーを簡単に作成し、維持できます。詳細については、外部ツールのドキュメントを参照してください。

AWS KMS 例外とエラーメッセージ

AWS KMS は、発生した問題に関する詳細なエラーメッセージを提供します。AWS KMS 例外の詳細については、[AWS Key Management Service API リファレンス](#)および AWS SDKs を参照してください。AWS KMS コンソールを使用している場合でも、これらのリファレンスが役立つ場合があります。例えば、CreateCustomKeyStores オペレーションの場合は、[エラー](#)リストを参照してください。

外部キーストアプロキシのパフォーマンスを最適化するため、AWS KMS は、5 分間の集約時間内に、お使いのプロキシの信頼性に基づく例外を返します。500 Internal Server Error、503 Service Unavailable、または接続タイムアウトが発生した場合、信頼性の高いプロキシであれば KMSInternalException を返し、自動再試行をトリガーして、リクエストが最終的に成功するようにします。一方で、信頼性の低いプロキシは KMSInvalidStateException を返します。詳細については、「[外部キーストアのモニタリング](#)」を参照してください。

外部キーストアの KMS キーを使用して別の AWS サービスのリソースを保護するなど、別の AWS サービスで問題が発生した場合、AWS そのサービスは問題の特定に役立つ追加情報を提供することがあります。AWS サービスがメッセージを提供しない場合は、KMS キーの使用を記録する [CloudTrail ログ](#) にエラーメッセージを表示できます。

CloudTrail ログ

AWS KMS コンソールのアクションを含むすべての AWS KMS API オペレーションは、AWS CloudTrail logs に記録されます。は、成功したオペレーションと失敗したオペレーションのログエントリ AWS KMS を記録します。オペレーションが失敗した場合、ログエントリには AWS KMS の例外名 (errorCode) とエラーメッセージ (errorMessage) が含まれます。この情報を使用してエラーを特定し、解決できます。例については、[外部キーストアの KMS キーを使用した復号の失敗](#)を参照してください。

ログエントリにはリクエスト ID も含まれます。リクエストが外部キーストアプロキシに到達した場合は、ログエントリのリクエスト ID を使用して、プロキシログで対応するリクエストを検索できます (プロキシが提供している場合)。

CloudWatch メトリクス

AWS KMS は、レイテンシー、スロットリング、プロキシエラー、外部キーマネージャーのステータス、TLS 証明書の有効期限が切れるまでの日数、プロキシ認証情報の報告された経過時間など、外部キーストアのオペレーションとパフォーマンスに関する詳細な Amazon CloudWatch メトリクスを記録します。これらのメトリクスを使用して、外部キーストアのオペレーションに関するデータモデルや、差し迫った問題が発生する前に警告する CloudWatch アラームを開発できます。

Important

AWS KMS では、CloudWatch アラームを作成して外部キーストアメトリクスをモニタリングすることをお勧めします。これらのアラームは、問題が発生する前に問題の初期兆候を警告します。

モニタリンググラフ

AWS KMS は、AWS KMS コンソールの各外部キーストアの詳細ページに外部キーストア CloudWatch メトリクスのグラフを表示します。グラフのデータは、エラー原因の特定、差し迫った問題の検出、ベースラインの確立、CloudWatch アラームのしきい値の調整に役立ちます。モニタリンググラフの解釈とデータ使用方法の詳細については、「[外部キーストアをモニタリングする](#)」を参照してください。

外部キーストアと KMS キーの表示

AWS KMS は、AWS KMS コンソールの外部キーストアと [DescribeCustomKeyStores](#) および [DescribeKey](#) オペレーションへのレスポンスで、外部キーストアと [DescribeKey](#) キーに関する詳

細情報を表示します。これらの表示には、外部キーストアの[接続状態](#)や KMS キーに関連付けられている外部キーの ID など、トラブルシューティングに使用できる情報を含む、外部キーストアと KMS キーの特別なフィールドが含まれています。詳細については、「[外部キーストアを表示する](#)」を参照してください。

[XKS プロキシテストクライアント](#)

AWS KMS は、外部キーストアプロキシが[AWS KMS 外部キーストアプロキシ API 仕様](#)に準拠していることを検証するオープンソースのテストクライアントを提供します。このテストクライアントを使用して、外部キーストアプロキシの問題を特定し、解決できます。

設定エラー

外部キーストアを作成するときは、[プロキシ認証情報](#)、[プロキシ URI エンドポイント](#)、[プロキシ URI パス](#)、[VPC エンドポイントサービス名](#)など、外部キーストアの設定を構成するプロパティ値を指定します。がプロパティ値のエラー AWS KMS を検出すると、オペレーションは失敗し、障害値を示すエラーを返します。

設定に関する問題の多くは、誤った値を修正することで解決できます。無効なプロキシ URI パスやプロキシ認証情報は、外部キーストアを切断せずに修正できます。一意性の要件を含む、これらの値の定義については「[前提条件を構成する](#)」を参照してください。これらの値を更新する手順については、「[外部キーストアプロパティを編集する](#)」を参照してください。

プロキシ URI パスとプロキシ認証情報値のエラーを避けるために、外部キーストアを作成または更新するときは、[プロキシ設定ファイル](#)を AWS KMS コンソールにアップロードします。これは、外部キーストアプロキシまたは外部キーマネージャーから提供される、プロキシ URI パスとプロキシ認証情報値を含む JSON ベースのファイルです。AWS KMS API オペレーションでプロキシ設定ファイルは使用できませんが、ファイルの値を使用して、プロキシの値と一致する API リクエストのパラメータ値を指定することができます。

一般的な設定エラー

例外: CustomKeyStoreInvalidStateException

(CreateKey)、KMSInvalidStateException (暗号化オペレーショ

ン)、XksProxyInvalidConfigurationException (管理オペレーション、CreateKey を除く)

[接続エラーコード:](#)

XKS_PROXY_INVALID_CONFIGURATION、XKS_PROXY_INVALID_TLS_CONFIGURATION

[パブリックエンドポイント接続](#)を持つ外部キーストアの場合、は外部キーストアを作成および更新するときにプロパティ値を AWS KMS テストします。[VPC エンドポイントサービス接続](#)を備えた外部キーストアの場合、AWS KMS は、外部キーストアの接続および更新時にプロパティ値をテストします。

 Note

外部キーストアをその外部キーストアのプロキシに接続する試みが失敗しても、非同期の ConnectCustomKeyStore オペレーションは成功する場合があります。この場合、例外はありませんが、外部キーストアの接続状態は「失敗」で、エラーメッセージを説明する接続エラーコードが表示されます。詳細については、「[外部キーストア接続エラー](#)」を参照してください。

がプロパティ値のエラー AWS KMS を検出すると、オペレーションは失敗し、次のいずれかのエラーメッセージ XksProxyInvalidConfigurationException が返されます。

URI パスが無効なため、外部キーストアプロキシがリクエストを拒否しました。外部キーストアの URI パスを検証し、必要に応じて更新します。

- [プロキシ URI パス](#)は、プロキシ API への AWS KMS リクエストのベースパスです。APIs このパスが間違っていると、プロキシへのリクエストはすべて失敗します。外部キーストアの[現在のプロキシ URI パスを表示する](#)には、AWS KMS コンソールまたは DescribeCustomKeyStores オペレーションを使用します。正しいプロキシ URI パスを検出するには、外部キーストアプロキシのドキュメントを参照してください。プロキシ URI パス値の修正については、「[外部キーストアプロパティを編集する](#)」を参照してください。
- 外部キーストアプロキシのプロキシ URI パスは、外部キーストアプロキシまたは外部キーマネージャーの更新によって変更される可能性があります。これらの変更については、外部キーストアプロキシまたは外部キーマネージャーのドキュメントを参照してください。

XKS_PROXY_INVALID_TLS_CONFIGURATION

AWS KMS が、外部キーストアプロキシへの TLS 接続を確立できません。証明書を含む、TLS 設定を検証します。

- すべての外部キーストアプロキシには TLS 証明書が必要です。TLS 証明書は、外部キーストアでサポートされている公開認証機関 (CA) によって発行される必要があります。サポートされている CA のリストについては、「AWS KMS 外部キーストアプロキシ API 仕様」の、「[信頼できる認証機関](#)」を参照してください。
- パブリックエンドポイント接続の場合、TLS 証明書のサブジェクト共通名 (CN) が、外部キーストアプロキシの[プロキシ URI エンドポイント](#)のドメイン名と一致する必要があります。例えば、パブリックエンドポイントが `https://myproxy.xks.example.com` の場合、TLS 証明書の CN は、`myproxy.xks.example.com` または `*.xks.example.com` である必要があります。
- VPC エンドポイントサービス接続の場合、TLS 証明書のサブジェクト共通名 (CN) が、[VPC エンドポイントサービス](#)のプライベート DNS 名と一致する必要があります。例えば、プライベート DNS 名が `myproxy-private.xks.example.com` の場合、TLS 証明書の CN は、`myproxy-private.xks.example.com` または `*.xks.example.com` である必要があります。
- TLS 証明書を有効期限切れにすることはできません。TLS 証明書の有効期限を取得するには、[OpenSSL](#) などの SSL ツールを使用します。外部キーストアに関連付けられた TLS 証明書の有効期限をモニタリングするには、[XksProxyCertificateDaysToExpire](#) CloudWatch メトリクスを使用します。TLS 証明書の有効期限が切れるまでの日数は、AWS KMS コンソール[のモニタリングセクション](#)にも表示されます。
- [パブリックエンドポイント接続](#)を使用している場合は、SSL テストツールを使用して SSL 設定をテストします。TLS 接続エラーは、証明書チェーンが間違っていることが原因である可能性があります。

VPC エンドポイントサービス接続設定のエラー

例外:

`XksProxyVpcEndpointServiceNotFoundException`、`XksProxyVpcEndpointServiceInvalidCo`

一般的な接続の問題に加えて、VPC エンドポイントサービス接続を使用して外部キーストアを作成、接続、または更新するときに、次の問題が発生することがあります。は、外部キーストア[を作成、接続、および更新](#)するときに、VPC エンドポイントサービス接続を使用して外部キーストアのプロパティ値を AWS KMS テストします。設定エラーが原因で管理オペレーションが失敗すると、次の例外が生成されます。

```
XksProxyVpcEndpointServiceNotFoundException
```

原因は、次のいずれかである可能性があります。

- VPC エンドポイントのサービス名が間違っています。外部キーストアの VPC エンドポイント サービス名が正しく、外部キーストアのプロキシ URI エンドポイント値と一致していることを検証します。VPC エンドポイントサービス名を検出するには、[Amazon VPC コンソール](#)または [DescribeVpcEndpointServices](#) オペレーションを使用します。既存の外部キーストアの VPC エンドポイントサービス名とプロキシ URI エンドポイントを検索するには、AWS KMS コンソールまたは [DescribeCustomKeyStores](#) オペレーションを使用します。詳細については、「[外部キーストアを表示する](#)」を参照してください。
- VPC エンドポイントサービスは、外部キーストア AWS リージョン とは異なる にある可能性があります。VPC エンドポイントサービスと外部キーストアが同じリージョンにあることを検証します。(リージョン名の外部名 (us-east-1 など) は、com.amazonaws.vpce.us-east-1.vpce-svc-example などの VPC エンドポイントサービス名の一部です)。外部キーストアの VPC エンドポイントサービス要件のリストについては、「[VPC エンドポイントサービス](#)」を参照してください。VPC エンドポイントサービスまたは外部キーストアを別のリージョンに移動することはできません。ただし、VPC エンドポイントサービスと同じリージョンに新しい外部キーストアを作成できます。詳細については、「[VPC エンドポイントサービス接続を設定する](#)」および「[外部キーストアを作成する](#)」を参照してください。
- AWS KMS は VPC エンドポイントサービスの許可されたプリンシパルではありません。VPC エンドポイントサービスの [Allow principals] (許可プリンシパル) リストには、cks.kms.eu-west-3.amazonaws.com のような cks.kms.<region>.amazonaws.com 値を含める必要があります。この値を追加する手順については、『AWS PrivateLink ガイド』の「[許可の管理](#)」を参照してください。

XksProxyVpcEndpointServiceInvalidConfigurationException

このエラーは、VPC エンドポイントサービスが次の要件のいずれかを満たしていない場合に発生します。

- VPC には、それぞれが異なるアベイラビリティーゾーンに存在する 2 つ以上のプライベートサブネットが必要です。VPC へのサブネットの追加の詳細については、『Amazon VPC ユーザーガイド』の「[VPC でのサブネットの作成](#)」を参照してください。
- [VPC エンドポイントのサービスタイプ](#)には、ゲートウェイロードバランサーではなく、Network Load Balancer を使用する必要があります。
- VPC エンドポイントサービスに承認は不要です (承認が必要は誤りです)。各接続リクエストの手動承認が必要な場合は、VPC エンドポイントサービスを使用して外部キーストアプロキシに接続

AWS KMS することはできません。詳細については、『AWS PrivateLink ガイド』の「[接続リクエストの承認または拒否](#)」を参照してください。

- VPC エンドポイントサービスには、パブリックドメインのサブドメインであるプライベート DNS 名が必要です。例えば、プライベート DNS 名が `https://myproxy-private.xks.example.com` の場合、`xks.example.com` または `example.com` ドメインにはパブリック DNS サーバーが必要です。VPC エンドポイントサービスのプライベート DNS 名を表示または変更するには、『AWS PrivateLink ガイド』の「[VPC エンドポイントサービスの DNS 名の管理](#)」を参照してください。
- プライベート DNS 名ドメインのドメイン検証ステータスは、`verified` である必要があります。プライベート DNS 名ドメインの検証ステータスを表示、更新するには、「[ステップ 5: プライベート DNS 名ドメインを検証する](#)」を参照して下さい。必要なテキストレコードを追加した後、更新された検証ステータが表示されるまでに数分かかる場合があります。

Note

プライベート DNS ドメインは、パブリックドメインのサブドメインである場合にのみ検証できます。サブドメインでない場合は、必要な TXT レコードを追加後も、プライベート DNS ドメインの検証ステータスは変わりません。

- AWS KMS と外部キーストアプロキシ間のファイアウォールで、プロキシのポート 443 との間のトラフィックが許可されていることを確認します。は、IPv4 経由でポート 443 で AWS KMS 通信します。この値は設定できません。
- VPC エンドポイントサービスのプライベート DNS 名は、外部キーストアの[プロキシ URI エンドポイント](#)値と一致する必要があります。VPC エンドポイントサービス接続を備えた外部キーストアの場合、プロキシ URI エンドポイントは `https://` で、その後に VPC エンドポイントサービスのプライベート DNS 名が続く必要があります。プロキシ URI エンドポイント値を表示するには、「[外部キーストアを表示する](#)」を参照してください。プロキシ URI エンドポイント値を変更するには、「[外部キーストアプロパティを編集する](#)」を参照してください。

外部キーストア接続エラー

[外部キーストアを外部キーストアプロキシに接続するプロセス](#)は、完了までに約 5 分かかります。すぐに失敗しない限り、ConnectCustomKeyStore オペレーションは HTTP 200 レスポンスと、プロパティを含まない JSON オブジェクトを返します。ただし、この初期レスポンスは接続に成功したことを示していません。外部キーストアが接続されているかどうかを判断するには、その[接続状](#)

態を参照してください。接続が失敗した場合、外部キーストアの接続状態は `FAILED` に変わり、失敗の原因を説明する [接続エラーコード](#) `FAILED AWS KMS` を返します。

Note

カスタムキーストアの接続状態が `FAILED` の場合は、再接続を試みる前にカスタムキーストアを切断する必要があります。 `FAILED` 接続ステータスでカスタムキーストアに接続することはできません。

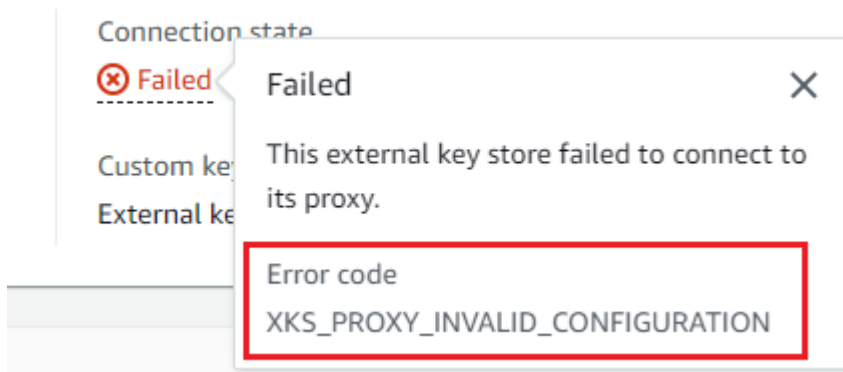
外部キーストアの接続状態を表示するには:

- [DescribeCustomKeyStores](#) レスポンスで、`ConnectionState` 要素の値を表示します。
- AWS KMS コンソールでは、接続状態が外部キーストアテーブルに表示されます。また、各外部キーストアの詳細ページの [General configuration] (一般設定) セクションに、[Connection state] (接続状態) が表示されます。

接続状態が `FAILED` の場合、接続エラーコードがエラーを説明します。

接続エラーコードを表示するには:

- [DescribeCustomKeyStores](#) レスポンスで、`ConnectionErrorCode` 要素の値を表示します。この要素は、`ConnectionState` が `FAILED` の場合にのみ、`DescribeCustomKeyStores` レスポンスに表示されます。
- AWS KMS コンソールで接続エラーコードを表示するには、外部キーストアの詳細ページで、失敗した値にカーソルを合わせます。



外部キーストアの接続エラーコード

次の接続エラーコードは、外部キーストアに適用されます

INTERNAL_ERROR

AWS KMS は、内部エラーのためリクエストを完了できませんでした。リクエストを再試行します。ConnectCustomKeyStore リクエストの場合、カスタムキーストアの接続を切断してから接続を再実行します。

INVALID_CREDENTIALS

指定された外部キーストアプロキシでは、XksProxyAuthenticationCredential 値の一方または両方が無効です。

NETWORK_ERRORS

ネットワークエラーにより、はカスタムキーストアをバックアップキーストアに接続 AWS KMS できません。

XKS_PROXY_ACCESS_DENIED

AWS KMS リクエストは、外部キーストアプロキシへのアクセスを拒否されます。外部キーストアプロキシに承認ルールがある場合は、そのルールがユーザーに代わって、AWS KMS とプロキシの通信を許可していることを検証します。

XKS_PROXY_INVALID_CONFIGURATION

設定エラーにより、外部キーストアはプロキシに接続できません。XksProxyUriPath の値を検証します。

XKS_PROXY_INVALID_RESPONSE

AWS KMS は、外部キーストアプロキシからのレスポンスを解釈できません。この接続エラーコードが繰り返し表示される場合は、外部キーストアプロキシベンダーに通知してください。

XKS_PROXY_INVALID_TLS_CONFIGURATION

AWS KMS TLS 設定が無効であるため、は外部キーストアプロキシに接続できません。外部キーストアプロキシが TLS 1.2 または 1.3 をサポートしていることを検証します。また、TLS 証明書の有効期限が切れていないこと、XksProxyUriEndpoint 値のホスト名と一致していること、[信頼できる認証機関](#)リストに含まれる信頼できる認証機関によって署名されていることを検証します。

XKS_PROXY_NOT_REACHABLE

AWS KMS は外部キーストアプロキシと通信できません。XksProxyUriEndpoint と XksProxyUriPath が正しいことを検証します。外部キーストアプロキシのツールを使用して、プロキシがアクティブで、ネットワーク上で使用可能であることを検証します。また、外部キーマネージャーインスタンスが正しく動作していることを検証します。外部キーマネージャーインスタンスがすべて使用できないとプロキシが報告した場合、接続試行はこの接続エラーコードで失敗します。

XKS_PROXY_TIMED_OUT

AWS KMS は外部キーストアプロキシに接続できますが、プロキシは割り当てられた時間 AWS KMS 内に に応答しません。この接続エラーコードが繰り返し表示される場合は、外部キーストアプロキシベンダーに通知してください。

XKS_VPC_ENDPOINT_SERVICE_INVALID_CONFIGURATION

Amazon VPC エンドポイントサービス設定が、AWS KMS 外部キーストアの要件に準拠していません。

- VPC エンドポイントサービスは、呼び出し元の AWS アカウントのインターフェイスエンドポイント用のエンドポイントサービスである必要があります。
- Network Load Balancer (NLB) は、それぞれが異なるアベイラビリティーゾーンに存在する 2 つ以上のサブネットに接続されていること。
- Allow principals リストには、cks.kms.<region>.amazonaws.comなどのリージョン AWS KMS のサービスプリンシパルが含まれている必要がありますcks.kms.us-east-1.amazonaws.com。
- 接続リクエストの[承認](#)を要求しないでください。
- プライベート DNS 名を付ける必要があります。VPC_ENDPOINT_SERVICE 接続の外部キーストアのプライベート DNS 名は、その AWS リージョン内で一意である必要があります。
- プライベート DNS 名のドメインの[検証ステータス](#)は、verified である必要があります。
- [TLS 証明書](#)は、エンドポイントにアクセス可能なプライベート DNS ホスト名を指定します。

XKS_VPC_ENDPOINT_SERVICE_NOT_FOUND

AWS KMS は、外部キーストアプロキシとの通信に使用する VPC エンドポイントサービスを見つけることができません。XksProxyVpcEndpointServiceName が正しく、AWS KMS サービスプリンシパルに、Amazon VPC エンドポイントサービスに対するサービスコンシューマー許可があることを検証します。

レイテンシーとタイムアウトエラー

例外: CustomKeyStoreInvalidStateException

(CreateKey)、KMSInvalidStateException (暗号化オペレーション)、XksProxyUriUnreachableException (管理オペレーション)

接続エラーコード: XKS_PROXY_NOT_REACHABLE、XKS_PROXY_TIMED_OUT

AWS KMS が 250 ミリ秒のタイムアウト期間内にプロキシに接続できない場合、例外が返されます。CreateCustomKeyStoreと UpdateCustomKeyStoreが返されますXksProxyUriUnreachableException。暗号化オペレーションは、問題を説明するエラーメッセージとともに標準 KMSInvalidStateException を返します。失敗ConnectCustomKeyStoreした場合、は問題を説明する[接続エラーコード](#) AWS KMS を返します。

タイムアウトエラーは一時的な問題のため、リクエストを再試行することで解決できる場合があります。問題が解決しない場合は、外部キーストアプロキシがアクティブでネットワークに接続されていること、そのプロキシ URI エンドポイント、プロキシ URI パス、VPC エンドポイントのサービス名 (存在する場合) が外部キーストアで正しいことを検証します。また、外部キーマネージャーが外部キーストア AWS リージョン のに近いことを確認します。これらの値のいずれかを更新する必要がある場合は、「[外部キーストアプロパティを編集する](#)」を参照して下さい。

レイテンシーパターンを追跡するには、AWS KMS コンソールの [\[Monitoring\] \(モニタリング\) セクション](#)にある [XksProxyLatency](#) CloudWatch メトリクスと (そのメトリクスに基づく) [Average latency] (平均レイテンシー) グラフを使用します。外部キーストアプロキシは、レイテンシーとタイムアウトを追跡するログやメトリクスも生成する場合があります。

XksProxyUriUnreachableException

AWS KMS は外部キーストアプロキシと通信できません。これは一時的なネットワークの問題である可能性があります。このエラーが繰り返し表示される場合は、外部キーストアプロキシがアクティブでネットワークに接続されていること、エンドポイント URI が外部キーストアで正しいことを検証します。

- 外部キーストアプロキシは、250 ミリ秒のタイムアウト期間内に AWS KMS プロキシ API リクエストに応答しませんでした。これは、一時的なネットワークの問題か、プロキシのオペレーションまたはパフォーマンスの問題である可能性があります。再試行しても問題が解決しない場合は、外部キーストアプロキシ管理者に通知してください。

多くの場合、レイテンシーやタイムアウトエラーは接続障害として表示されま

す。[ConnectCustomKeyStore](#) オペレーションが失敗すると、外部キーストアの接続状態が `FAILED` に変わり、エラーを説明する接続エラーコード `FAILED_AWS_KMS` が返されます。接続エラーコードのリストとエラー解決方法については、「[外部キーストアの接続エラーコード](#)」を参照してください。[All custom key stores] (すべてのカスタムキーストア) と [External key stores] (外部キーストア) の接続コードリストは、外部キーストアに適用されます。次の接続エラーは、レイテンシーとタイムアウトに関連しています。

`XKS_PROXY_NOT_REACHABLE`

-または-

`CustomKeyStoreInvalidStateException` , `KMSInvalidStateException` ,
`XksProxyUriUnreachableException`

AWS KMS は外部キーストアプロキシと通信できません。外部キーストアプロキシがアクティブでネットワークに接続されていること、外部キーストアの URI パスとエンドポイント URI または VPC サービス名が正しいことを検証します。

このエラーは、次の原因によって発生する可能性があります。

- 外部キーストアプロキシがアクティブでないか、ネットワークに接続されていない。
- 外部キーストア設定の[プロキシ URI エンドポイント](#)、[プロキシ URI パス](#)、または [VPC エンドポイントのサービス名](#) (該当する場合) の値にエラーがある。外部キーストアの設定を表示するには、[DescribeCustomKeyStores](#) オペレーションを使用するか、[コンソールで外部キーストアの詳細ページを表示します](#)。AWS KMS
- AWS KMS と外部キーストアプロキシとの間のネットワークパスに、ポートエラーなどのネットワーク設定エラーが発生する可能性があります。はIPv4 経由でポート 443 の外部キーストアプロキシと AWS KMS 通信します。この値は設定できません。
- ([GetHealthStatus](#) レスポンスで) 外部キーストアプロキシが、すべての外部キーマネージャーインスタンスが `UNAVAILABLE` であると報告すると、[ConnectCustomKeyStore](#) オペレーションは `XKS_PROXY_NOT_REACHABLE` の `ConnectionErrorCode` で失敗します。ヘルプについては、外部キーマネージャーのドキュメントを参照してください。
- このエラーは、外部キーマネージャーと外部キーストア AWS リージョン を持つ との物理的な距離が長いことが原因で発生する可能性があります。AWS リージョン と外部キーマネージャー間の ping レイテンシー (ネットワークラウンドトリップタイム (RTT)) は 35 ミリ秒以下にする必要

があります。外部キーマネージャー AWS リージョン に近い に外部キーストアを作成するか、外部キーマネージャーを に近いデータセンターに移動する必要があります AWS リージョン。

XKS_PROXY_TIMED_OUT

-または-

CustomKeyStoreInvalidStateException , KMSInvalidStateException ,
XksProxyUriUnreachableException

外部キーストアプロキシが時間内に応答しなかったため、AWS KMS はリクエストを拒否しました。リクエストを再試行します。このエラーが繰り返し表示される場合は、外部キーストアプロキシ管理者に報告してください。

このエラーは、次の原因によって発生する可能性があります。

- このエラーは、外部キーマネージャーと外部キーストアプロキシ間の物理的な距離が長いために発生する可能性があります。可能な場合は、外部キーストアプロキシを外部キーマネージャーの近くに移動します。
- タイムアウトエラーは、プロキシがリクエストのボリュームと頻度を処理するように設計されていない場合に発生する可能性があります AWS KMS。CloudWatch メトリクスが持続的な問題を示している場合は、外部キーストアプロキシ管理者に通知してください。
- タイムアウトエラーは、外部キーマネージャーと外部キーストアの Amazon VPC 間の接続が正しく機能していない場合に発生する可能性があります。を使用している場合は AWS Direct Connect、VPC と外部キーマネージャーが効果的に通信できることを確認します。問題の解決方法については、「AWS Direct Connect ユーザーガイド」の [「トラブルシューティング AWS Direct Connect」](#) を参照してください。

XKS_PROXY_TIMED_OUT

-または-

CustomKeyStoreInvalidStateException , KMSInvalidStateException ,
XksProxyUriUnreachableException

外部キーストアプロキシが割り当てられた時間内にリクエストに応答しません。リクエストを再試行します。このエラーが繰り返し表示される場合は、外部キーストアプロキシ管理者に報告してください。

- このエラーは、外部キーマネージャーと外部キーストアプロキシ間の物理的な距離が長いために発生する可能性があります。可能な場合は、外部キーストアプロキシを外部キーマネージャーの近くに移動します。

認証情報エラー

例外: CustomKeyStoreInvalidStateException

(CreateKey)、KMSInvalidStateException (暗号化オペレーシヨ

ン)、XksProxyIncorrectAuthenticationCredentialException (CreateKey 以外の管理オペレーション)

外部キーストアプロキシ AWS KMS で の認証情報を確立して維持します。次に AWS KMS、外部キーストアを作成するときに認証情報値を指定します。認証情報を変更する場合は、外部キーストアプロキシで変更します。次に、外部キーストアの[認証情報を更新します](#)。プロキシが認証情報をローテーションする場合は、外部キーストアの[認証情報を更新する](#)必要があります。

外部キーストアプロキシが、外部キーストアの[プロキシ認証情報](#)で署名されたリクエストを認証しない場合、その影響はリクエストによって異なります。

- CreateCustomKeyStore および UpdateCustomKeyStore は XksProxyIncorrectAuthenticationCredentialException で失敗します。
- ConnectCustomKeyStore は成功するが接続は失敗します。接続状態は FAILED で、接続エラーコードは INVALID_CREDENTIALS です。詳細については、「[外部キーストア接続エラー](#)」を参照してください。
- 外部キーストアのすべての外部設定エラーと接続状態エラーに対して、暗号化オペレーションは KMSInvalidStateException を返します。添付のエラーメッセージは問題を説明しています。

外部キーストアプロキシは AWS KMSを認証できなかったため、リクエストを拒否しました。外部キーストアの認証情報を検証し、必要に応じて更新します。

このエラーは、次の原因によって発生する可能性があります。

- 外部キーストアのアクセスキー ID またはシークレットアクセスキーが、外部キーストアプロキシに設定された値と一致しない。

このエラーを修正するには、外部キーストアの[プロキシ認証情報を更新します](#)。この変更は、外部キーストアを切断せずに行うことができます。

- AWS KMS と外部キーストアプロキシ間のリバースプロキシは、SigV4 署名を無効にする方法で HTTP ヘッダーを操作する可能性があります。このエラーを修正するには、プロキシ管理者に通知してください。

キーステータスエラー

例外: `KMSInvalidStateException`

`KMSInvalidStateException` は、カスタムキーストアの KMS キーに、2 つの異なる目的で使われます。

- `CancelKeyDeletion` などの管理オペレーションが失敗してこの例外が返された場合は、KMS キーの[キーステータス](#)にオペレーションとの互換性がないことを示します。
- カスタムキーストアの KMS キーに対する[暗号化オペレーション](#)が `KMSInvalidStateException` で失敗した場合、KMS キーのキーステータスに問題がある可能性があります。ただし、AWS KMS 暗号化オペレーション `KMSInvalidStateException` は、外部キーストアのすべての外部設定エラーと接続状態エラーに対してを返します。問題を特定するには、例外に付随するエラーメッセージを使用します。

AWS KMS API オペレーションに必要なキーの状態を確認するには、「」を参照してください[キーの AWS KMS キーステータス](#)。KMS キーのキーステータスを確認するには、[Customer managed keys] (カスタマーマネージドキー) ページで、KMS キーの [Status] (ステータス) フィールドを表示します。または、[DescribeKey](#) オペレーションを使用して、レスポンスの `KeyState` 要素を表示します。詳細については、「[キーの特定と表示](#)」を参照してください。

Note

外部キーストアの KMS キーのキーステータスは、関連付けられている[外部キー](#)のステータスを示すものではありません。外部キーステータスに関する情報は、外部キーマネージャーと外部キーストアプロキシツールを参照してください。

`CustomKeyStoreInvalidStateException` は、KMS キーの[キーステータス](#)ではなく、外部キーストアの[接続状態](#)を参照します。

KMS キーのキーステータスが Unavailable または PendingDeletion の場合、カスタムストアの KMS キーの暗号化オペレーションは失敗する可能性があります。(無効化されたキーは DisabledException を返します)。

- KMS キーは、AWS KMS コンソールで意図的に KMS キーを無効にするか、[DisableKey](#) オペレーションを使用する場合にのみキーDisabled状態になります。KMS キーが無効になっている間、キーを表示および管理することはできますが、暗号化オペレーションで使用することはできません。この問題を解決するには、キーを有効にします。詳細については、「[キーの有効化と無効化](#)」を参照してください。
- KMS キーは、外部キーストアが外部キーストアプロキシから切断されたときに Unavailable キーステータスになります。使用できない KMS キーを修正するには、[外部キーストアを再接続します](#)。外部キーストアを再接続すると、外部キーストア内の KMS キーのキーステータスは、Enabled や Disabled などの以前のステータスに自動的に復元されます。

KMS キーは、削除予定の待機期間中、PendingDeletion キーステータスになります。削除保留中の KMS キーのキーステータスエラーは、そのキーが暗号化に使用されているか、復号に必要であるため、キーを削除するべきではないことを示します。KMS キーを再度有効にするには、スケジュールされた削除をキャンセルしてから、[キーを有効にします](#)。詳細については、「[キー削除をスケジュールする](#)」を参照してください。

復号エラー

例外: KMSInvalidStateException

外部キーストアの KMS キーを使用した[復号](#)オペレーションが失敗すると、は、外部キーストアのすべての外部設定エラーと接続状態エラーに暗号化オペレーションKMSInvalidStateExceptionが使用する標準 AWS KMS を返します。問題を示すエラーメッセージ。

外部キーマネージャーは、[二重暗号化](#)を使用して暗号化された暗号文を復号するために、最初に外部キーを使用して暗号文の外側のレイヤーを復号します。次に AWS KMS、は KMS AWS KMS キーのキーマテリアルを使用して暗号文の内部レイヤーを復号します。無効、または破損した暗号文は、外部キーマネージャーまたは AWS KMSによって拒否される可能性があります。

復号に失敗すると、次のエラーメッセージが KMSInvalidStateException に付随します。このエラーメッセージは、リクエスト内の暗号文またはオプションの暗号化コンテキストに問題があることを示します。

指定された暗号文または追加の認証データが破損している、欠損している、または無効であるため、外部キーストアプロキシはリクエストを拒否しました。

- 外部キーストアプロキシまたは外部キーマネージャーが暗号文またはその暗号化コンテキストが無効であると報告した場合、通常、送信されたDecryptリクエストの暗号文または暗号化コンテキストに問題があることを示します AWS KMS。Decrypt オペレーションの場合、はDecryptリクエストで受信したのと同じ暗号文と暗号化コンテキストをプロキシ AWS KMS に送信します。

このエラーは、ビットの反転など、転送中のネットワークの問題が原因である可能性があります。Decrypt リクエストを再試行します。問題が解決しない場合は、暗号文が変更されていないこと、または破損していないことを検証します。また、Decryptリクエストの暗号化コンテキストが、データを暗号化したリクエストの暗号化コンテキスト AWS KMS と一致することを確認します。

外部キーストアプロキシが復号のために送信した暗号文、または暗号化コンテキストが、破損しているか、欠損しているか、無効になっています。

- がプロキシから受信した暗号文 AWS KMS を拒否すると、外部キーマネージャーまたはプロキシが無効な暗号文または破損した暗号文を返したことを示します AWS KMS。

このエラーは、ビットの反転など、転送中のネットワークの問題が原因である可能性があります。Decrypt リクエストを再試行します。問題が解決しない場合は、外部キーマネージャーが正しく動作していること、および外部キーストアプロキシが外部キーマネージャーから受け取った暗号文を返す前に変更しないことを確認します AWS KMS。

外部キーエラー

[外部キー](#)は、KMS キーの外部キーマテリアルとして機能する外部キーマネージャーの暗号化キーです。AWS KMS は外部キーには直接アクセスできません。外部キーマネージャーに、(外部キーストアプロキシ経由で) 外部キーを使用してデータを暗号化するか、暗号文を復号するように要求する必要があります。

外部キーストアで KMS キーを作成する際に、外部キーマネージャーで外部キーの ID を指定します。KMS キー作成後に外部キー ID を変更することはできません。KMS キーに関する問題を防ぐために、CreateKey オペレーションは外部キーストアプロキシに、外部キーの ID と設定の検証を要

求します。外部キーが KMS キーでの使用 [要件を満たしていない](#)場合、CreateKey オペレーションは失敗し、問題を特定する例外とエラーメッセージが表示されます。

ただし、KMS キー作成後に問題が発生する可能性があります。外部キーの問題によって暗号化オペレーションが失敗すると、オペレーションは失敗し、問題を示すエラーメッセージとともに `KMSInvalidStateException` が返されます。

外部キーの CreateKey エラー

例外:

`XksKeyAlreadyInUseException`、`XksKeyNotFoundException`、`XksKeyInvalidConfigurationException`

[CreateKey](#) オペレーションは、[External key ID] (外部キー ID) (コンソール) または `XksKeyId` (API) パラメータで指定した外部キーの ID とプロパティを検証しようとします。このプラクティスは、KMS キーで外部キーを使用する前にエラーを早期に検出することを目的としています。

外部キー使用中

外部キーストアの各 KMS キーは、異なる外部キーを使用する必要があります。CreateKey は、KMS キーの外部キー ID (`XksKeyId`) が外部キーストアで一意ではないことを認識すると、`XksKeyAlreadyInUseException` により失敗します。

同じ外部キーに複数の ID を使用する場合、CreateKey は重複しているものを認識しません。ただし、同じ外部キーを持つ KMS キーは、AWS KMS キーマテリアルとメタデータが異なるため、相互運用できません。

外部キーを検出できない

外部キーストアプロキシが、KMS キーの外部キー ID (`XksKeyId`) を使用して外部キーを検出できないことを報告すると、CreateKey オペレーションは失敗し、次のエラーメッセージを含む `XksKeyNotFoundException` が返されます。

外部キーを検出できなかったため、外部キーストアプロキシはリクエストを拒否しました。

このエラーは、次の原因によって発生する可能性があります。

- KMS キーの外部キー (`XksKeyId`) ID が無効である可能性があります。外部キープロキシが外部キーを識別するために使用する ID を検出するには、外部キーストアプロキシまたは外部キーマネージャーのドキュメントを参照してください。

- 外部キーが外部キーマネージャーから削除された可能性があります。調査するには、外部キーマネージャーツールを使用します。外部キーが完全に削除された場合は、KMS キーで別の外部キーを使用します。外部キーのリストまたは要件については、「[外部キーストアの KMS キーの要件](#)」を参照してください。

外部キー要件が満たされていない

外部キーストアプロキシが、外部キーが KMS キーでの使用[要件を満たしていない](#)ことを報告すると、CreateKey オペレーションは失敗し、次のいずれかのエラーメッセージを含む XksKeyInvalidConfigurationException が返されます。

外部キーのキースペックは AES_256 である必要があります。指定された外部キーのキースペックは *<key-spec>* です。

- 外部キーは、AES_256 のキースペックをもつ 256 ビット対称暗号化キーである必要があります。指定された外部キーが別のタイプである場合は、この要件を満たす外部キー ID を指定します。

外部キーのステータスは ENABLED である必要があります。指定された外部キーのステータスは *<status>* です。

- 外部キーは、外部キーマネージャーで有効にする必要があります。指定した外部キーが有効になっていない場合は、外部キーマネージャーツールを使用して有効にするか、有効な外部キーを指定します。

外部キーのキー用途には、[ENCRYPT] と [DECRYPT] を含める必要があります。指定された外部キーのキー用途は *<key-usage>* です。

- 外部キーは、外部キーマネージャーで暗号化と復号を行うように設定する必要があります。指定された外部キーにこれらのオペレーションが含まれていない場合は、外部キーマネージャーツールを使用してオペレーションを変更するか、別の外部キーを指定します。

外部キーの暗号化オペレーションエラー

例外: KMSInvalidStateException

外部キーストアプロキシが KMS キーに関連付けられた外部キーを検出できない場合、または外部キーが KMS キーでの使用 [要件を満たしていない](#) 場合、暗号化オペレーションは失敗します。

暗号化オペレーション中に検出された外部キーの問題は、KMS キー作成前に検出された外部キーの問題よりも解決が困難です。KMS キー作成後に外部キー ID を変更することはできません。KMS キーでまだデータが暗号化されていない場合は、KMS キーを削除し、別の外部キー ID を使用して新しい KMS キーを作成できます。ただし、KMS キーで生成された暗号文は、キーのメタデータとキー AWS KMS マテリアルが異なるため、外部キーが同じであっても、他の KMS キーで復号することはできません。代わりに、外部キーマネージャツールを使用して、可能な限り外部キーの問題を解決してください。

外部キーストアプロキシが外部キーに関する問題を報告すると、暗号化オペレーションは、問題を特定するエラーメッセージとともに KMSInvalidStateException を返します。

外部キーを検出できない

外部キーストアプロキシが、KMS キーの外部キー ID (XksKeyId) を使用して外部キーを検出できないことを報告すると、暗号化オペレーションは次のエラーメッセージを含む KMSInvalidStateException を返します。

外部キーを検出できなかったため、外部キーストアプロキシはリクエストを拒否しました。

このエラーは、次の原因によって発生する可能性があります。

- KMS キーの外部キー (XksKeyId) ID が無効になりました。

KMS キーに関連付けられている外部キー ID を検出するには、[KMS キーの詳細を参照してください](#)。外部キープロキシが外部キーを識別するために使用する ID を検出するには、外部キーストアプロキシまたは外部キーマネージャのドキュメントを参照してください。

AWS KMS は、外部キーストアに KMS キーを作成するときに、外部キー ID を検証します。ただし、特に外部キー ID 値がエイリアスまたは可変名の場合は、ID が無効になる可能性があります。既存の KMS キーに関連付けられている外部キー ID を変更することはできません。KMS キーで暗号化された暗号文を復号するには、外部キーを既存の外部キー ID に再度、関連付ける必要があります。

KMS キーを使用してデータを暗号化していない場合は、有効な外部キー ID を使用して新しい KMS キーを作成できます。ただし、KMS キーを使用して暗号文を生成した場合は、同じ外部キーを使用しても、他の KMS キーを使用して暗号文を復号することはできません。

- 外部キーが外部キーマネージャーから削除された可能性があります。調査するには、外部キーマネージャーツールを使用します。可能な場合は、外部キーマネージャーのコピーまたはバックアップから、[キーマテリアルの復元](#)を試みてください。外部キーが完全に削除された場合、関連する KMS キーで暗号化された暗号文は回復できません。

外部キー設定エラー

外部キーストアプロキシが、外部キーが KMS キーでの使用[要件を満たしていない](#)ことを報告すると、暗号化オペレーションは、次のいずれかのエラーメッセージを含む `KMSInvalidStateException` を返します。

外部キーがリクエストされたオペレーションをサポートしていないため、外部キーストアプロキシは要求を拒否しました。

- 外部キーは、暗号化と復号の両方をサポートしている必要があります。キーの用途に暗号化と復号が含まれていない場合は、外部キーマネージャーツールを使用して、キーの用途を変更します。

外部キーが外部キーマネージャーで有効になっていないため、外部キーストアプロキシはリクエストを拒否しました。

- 外部キーは、外部キーマネージャーで使用するために、有効にして使用可能にする必要があります。外部キーのステータスが `[Enabled]` でない場合は、外部キーマネージャーツールを使用して有効にします。

プロキシの問題

例外:

`CustomKeyStoreInvalidStateException (CreateKey)`、`KMSInvalidStateException` (暗号化オペレーション)

ン)、`UnsupportedOperationException`、`XksProxyUriUnreachableException`、`XksProxyInvalid` (CreateKey 以外の管理オペレーション)

外部キーストアプロキシは、AWS KMS と外部キーマネージャー間のすべての通信を仲介します。一般的な AWS KMS リクエストを、外部キーマネージャーが理解できる形式に変換します。外部キーストアプロキシが[AWS KMS 外部キーストアプロキシ API 仕様](#)に準拠していない場合、またはが正しく動作していない場合、またはと通信できない場合 AWS KMS、外部キーストアで KMS キーを作成または使用することはできません。

外部キーストアプロキシは外部キーストアアーキテクチャで重要な役割を果たすため、多くのエラーに外部キーストアプロキシが記載されますが、これらの問題は、外部キーマネージャーまたは外部キーに起因する可能性があります。

このセクションの問題は、外部キーストアプロキシの設計または運用に関する問題と関連しています。これらの問題を解決するには、プロキシソフトウェアを変更する必要がある場合があります。プロキシ管理者に相談してください。AWS KMS は、プロキシの問題の診断に役立つように、外部キーストアプロキシが[AWS KMS 外部キーストアプロキシ API 仕様](#)に準拠していることを検証する、オープンソースのテストクライアントである[XKS プロキシテキストクライアント](#)を提供しています。

`CustomKeyStoreInvalidStateException`、`KMSInvalidStateException`、または `XksProxyUriUnreachableException`

外部キーストアプロキシが異常な状態です。このメッセージが繰り返し表示される場合は、外部キーストアプロキシ管理者に通知してください。

- このエラーは、外部キーストアプロキシの運用上の問題またはソフトウェアエラーを示している可能性があります。各エラーを生成した AWS KMS API オペレーションの CloudTrail ログエントリを見つけることができます。このエラーは、オペレーションを再試行することで解決する場合があります。ただし、それでも解決しない場合は、外部キーストアプロキシ管理者に通知してください。
- 外部キーストアプロキシが、([GetHealthStatus](#) 応答で) すべての外部キーマネージャーインスタンスが UNAVAILABLE であると報告した場合、外部キーストアを作成または更新しようとする、この例外が発生して失敗します。このエラーが解決されない場合は、外部キーマネージャーのドキュメントを参照してください。

`CustomKeyStoreInvalidStateException`、`KMSInvalidStateException`、または `XksProxyInvalidResponseException`

AWS KMS は、外部キーストアプロキシからのレスポンスを解釈できません。このエラーが繰り返し表示される場合は、外部キーストアプロキシ管理者に相談してください。

- AWS KMS オペレーションは、プロキシが解析または解釈 AWS KMS できない未定義のレスポンスを返すと、この例外を生成します。このエラーは、一時的な外部の問題や散発的なネットワークエラーが原因で発生することがあります。ただし、問題が解決しない場合は、外部キーストアプロキシが、[AWS KMS 外部キーストアプロキシ API 仕様](#)に準拠していない可能性があります。外部キーストア管理者またはベンダーに通知してください。

`CustomKeyStoreInvalidStateException`、`KMSInvalidStateException`、または `UnsupportedOperationException`

リクエストされた暗号化オペレーションをサポートしていないため、外部キーストアプロキシはリクエストを拒否しました。

- 外部キーストアプロキシは、[AWS KMS 外部キーストアプロキシ API 仕様](#)で定義されているすべての [プロキシ API](#) をサポートしている必要があります。このエラーは、プロキシがリクエストに関連するオペレーションをサポートしていないことを示します。外部キーストア管理者またはベンダーに通知してください。

プロキシの承認に関する問題

例外: `CustomKeyStoreInvalidStateException`、`KMSInvalidStateException`

外部キーストアプロキシの中には、外部キーを使用するための認可要件を実装しているものがあります。外部キーストアプロキシは、特定のユーザーが任意の条件下で任意のオペレーションをリクエストすることを許可する承認スキームを設計および実装することを許可されていますが、必須ではありません。例えば、プロキシでは、ユーザーに任意の外部キーによる暗号化を許可しても、その外部キーによる復号は許可しない場合があります。詳細については、「[外部キーストアプロキシ認可 \(オペレーション\)](#)」を参照してください。

プロキシ認可は、プロキシへのリクエストに AWS KMS を含むメタデータに基づいています。awsSourceVpc および awsSourceVpce フィールドは、リクエストが VPC エンドポイントか

らのものである場合と、呼び出し側が KMS キーと同じアカウントにある場合にのみ、メタデータに含まれます。

```
"requestMetadata": {
  "awsPrincipalArn": string,
  "awsSourceVpc": string, // optional
  "awsSourceVpce": string, // optional
  "kmsKeyArn": string,
  "kmsOperation": string,
  "kmsRequestId": string,
  "kmsViaService": string // optional
}
```

認可の失敗によりプロキシがリクエストを拒否すると、関連する AWS KMS オペレーションは失敗します。は CreateKeyを返しますCustomKeyStoreInvalidStateException。AWS KMS 暗号化オペレーションは を返しますKMSInvalidStateException。どちらも以下のエラーメッセージを使用します。

外部キーストアプロキシはオペレーションへのアクセスを拒否しました。ユーザーと外部キーの両方がこのオペレーションを承認していることを検証し、リクエストを再試行します。

- エラーを解決するには、外部キーマネージャーまたは外部キーストアプロキシツールを使用して、承認が失敗した理由を特定します。次に、未承認リクエストの原因となった手順を更新するか、外部キーストアプロキシツールを使用して、承認ポリシーを更新します。AWS KMSではこのエラーを解決できません。

のセキュリティ AWS Key Management Service

でのクラウドセキュリティが最優先事項 AWS です。お客様は AWS、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、AWS お客様とお客様の間の責任共有です。[責任共有モデル](#)ではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。AWS また、では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、[AWS コンプライアンスプログラム](#)コンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。AWS Key Management Service (AWS KMS) に適用されるコンプライアンスプログラムの詳細については、「[コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム](#)」を参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。さらに AWS KMS、の設定と使用に加えて AWS KMS keys、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、を使用する際の責任共有モデルの適用方法を理解するのに役立ちます AWS Key Management Service。セキュリティとコンプライアンスの目的を達成する AWS KMS ように を設定する方法を示します。

トピック

- [でのデータ保護 AWS Key Management Service](#)
- [の ID とアクセスの管理 AWS Key Management Service](#)
- [でのログ記録とモニタリング AWS Key Management Service](#)
- [のコンプライアンス検証 AWS Key Management Service](#)
- [の耐障害性 AWS Key Management Service](#)
- [のインフラストラクチャセキュリティ AWS Key Management Service](#)

でのデータ保護 AWS Key Management Service

AWS Key Management Service は暗号化キーを保存および保護して可用性を高め、強力で柔軟なアクセスコントロールを提供します。

トピック

- [キーマテリアルの保護](#)
- [データ暗号化](#)
- [インターネットトラフィックのプライバシー](#)

キーマテリアルの保護

デフォルトでは、は KMS キーの暗号化キーマテリアル AWS KMS を生成して保護します。さらに、は、の外部で作成および保護されるキーマテリアルのオプション AWS KMS を提供します AWS KMS。

で生成されたキーマテリアルの保護 AWS KMS

KMS キーを作成すると、デフォルトでは KMS キーの暗号化マテリアル AWS KMS を生成して保護します。

KMS キーのキーマテリアルを保護するために、AWS KMS は [FIPS 140-3 セキュリティレベル 3 検証済み](#) ハードウェアセキュリティモジュール (HSMs)。各 AWS KMS HSM は、のセキュリティとスケーラビリティの要件を満たす専用の暗号化機能を提供するように設計された、専用のスタンドアロンハードウェアアプライアンスです AWS KMS。(中国リージョンで AWS KMS を使用する HSMs は [OSCCA](#) によって認定されており、関連するすべての中国の規制に準拠していますが、FIPS 140-3 暗号化モジュール検証プログラムでは検証されていません)。

KMS キーのキーマテリアルは、HSM で生成されるときにデフォルトで暗号化されます。キーマテリアルは HSM の揮発性メモリ内でのみ、暗号化オペレーションで使用するのにかかる数ミリ秒の間だけ復号化されます。キーマテリアルがアクティブに使用されていない場合は常に、HSM 内で暗号化され、[耐久性の高い](#) (99.999999999%)、低レイテンシーの永続ストレージに転送され、そこで HSM とは別の場所に保管されます。プレーンテキストのキーマテリアルは、HSM [セキュリティ境界](#) を離れることはありません。また、ディスクに書き込まれることも、ストレージメディアに保持されることもありません。(唯一の例外は、非対称キーペアのパブリックキーです。これはシークレットではありません)。

AWS は、基本的なセキュリティ原則として、どのタイプのプレーンテキスト暗号化キーマテリアルとも人間によるやり取りがないとアサートします AWS のサービス。AWS のサービス 演算子を含むすべてのユーザーがプレーンテキストのキーマテリアルを表示、アクセス、またはエクスポートするメカニズムはありません。この原則は、壊滅的な障害やディザスタリカバリ中にも適用されます。のプレーンテキストのカスタマーキーマテリアル AWS KMS は、FIPS AWS KMS 140-3 検証済み HSMs 内の暗号化オペレーションに使用され、顧客またはその代理人がサービスに対して行った承認されたリクエストにのみ応答します。

[カスタマーマネージドキー](#)の場合、キー AWS アカウント を作成する は、キーの唯一かつ譲渡不可能な所有者です。所有しているアカウントは、キーへのアクセスを制御する権限付与ポリシーを完全かつ排他的に制御できます。の場合 AWS マネージドキー、AWS アカウント は へのリクエストを承認する IAM ポリシーを完全に制御できます AWS のサービス。

AWS KMSの外部で生成されるキーマテリアルの保護

AWS KMS は、で生成されたキーマテリアルの代替手段を提供します AWS KMS。

オプションの AWS KMS 機能である[カスタムキーストア](#)を使用すると、外部で生成および使用されるキーマテリアルでバックアップされた KMS キーを作成できます AWS KMS。[AWS CloudHSM キーストア](#)の KMS キーは、ユーザーが管理する AWS CloudHSM ハードウェアセキュリティモジュールのキーによってバックアップされます。これらの HSMsは、[FIPS 140-2 セキュリティレベル 3 または 140-3 セキュリティレベル 3](#)で認定されています。[外部キーストアの KMS キー](#)は、プライベートデータセンターの物理 HSM など AWS、外部で制御および管理している外部キーマネージャーのキーによってバックアップされます。

もう 1 つのオプション機能により、KMS キーの[キーマテリアルをインポート](#)できます。への転送中にインポートされたキーマテリアルを保護するには AWS KMS、AWS KMS HSM で生成された RSA キーペアからパブリックキーを使用してキーマテリアルを暗号化します。インポートされたキーマテリアルは AWS KMS HSM で復号され、HSM の対称キーで再暗号化されます。すべての AWS KMS キーマテリアルと同様に、プレーンテキストでインポートされたキーマテリアルが HSMs を暗号化されないままにすることはありません。ただし、キーマテリアルを提供したお客様は、AWS KMSの外部におけるキーマテリアルの安全な使用、耐久性、メンテナンスに対して責任を持ちます。

データ暗号化

のデータは、AWS KMS keys とそれらが表す暗号化キーマテリアル AWS KMS で構成されます。このキーマテリアルは、AWS KMS ハードウェアセキュリティモジュール (HSM) 内でのみ、かつ使用

中の場合にのみ、プレーンテキストで存在します。それ以外の場合、キー素材は暗号化され、耐久性のある永続ストレージに保存されます。

が KMS キー用に AWS KMS 生成するキーマテリアルは HSMs の AWS KMS 境界を暗号化されないままにすることはありません。AWS KMS API オペレーションではエクスポートまたは送信されません。ただし、[マルチリージョンキー](#)は例外です。はクロスリージョンレプリケーションメカニズム AWS KMS を使用して、マルチリージョンキーのキーマテリアルを 1 つの HSM から別の の AWS リージョン HSM にコピーします AWS リージョン。詳細については、「暗号化の詳細」の「[マルチリージョンキーのレプリケーションプロセス](#)」を参照してください。AWS Key Management Service

トピック

- [保管中の暗号化](#)
- [転送中の暗号化](#)

保管中の暗号化

AWS KMS は、FIPS 140-3 セキュリティレベル 3 準拠のハードウェアセキュリティモジュール (HSMs) AWS KMS keys で のキーマテリアルを生成します。 <https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4884> 唯一の例外は中国リージョンで、 が KMS キーの生成 AWS KMS に使用する HSMs は、関連するすべての中国の規制に準拠していますが、FIPS 140-3 暗号化モジュール検証プログラムでは検証されていません。使用されていない場合、キーマテリアルは HSM キーによって暗号化され、耐久性のある永続的なストレージに書き込まれます。KMS キーのキーマテリアルおよびキーマテリアルを保護する暗号化キーは、HSM をプレーンテキスト形式のままにしません。

KMS キーのキーマテリアルの暗号化と管理は、AWS KMSによって完全に処理されます。

詳細については、「暗号化の詳細」の AWS Key Management Service 「[の使用 AWS KMS keys](#)」を参照してください。

転送中の暗号化

が KMS キー用に AWS KMS 生成するキーマテリアルは、AWS KMS API オペレーションではエクスポートまたは送信されません。は [キー識別子](#) AWS KMS を使用して、API オペレーションの KMS キーを表します。同様に、AWS KMS [カスタムキーストアの KMS キーのキー](#)マテリアルはエクスポートできず、AWS KMS または AWS CloudHSM API オペレーションでは送信されません。

ただし、一部の AWS KMS API オペレーションは[データキー](#)を返します。お客様は API オペレーションを使用して、選択した KMS キーの[キーマテリアルをインポート](#)することもできます。

すべての AWS KMS API コールは、Transport Layer Security (TLS) を使用して署名および送信する必要があります。は TLS 1.2 AWS KMS を必要とし、すべてのリージョンで TLS 1.3 を推奨します。は、中国リージョンを除くすべてのリージョン AWS KMS のサービスエンドポイントのハイブリッドポスト量子 TLS AWS KMS もサポートしています。AWS KMS は、の FIPS エンドポイントのハイブリッドポスト量子 TLS をサポートしていません AWS GovCloud (US)。AWS KMS の呼び出しには、PFS (Perfect Forward Secrecy) をサポートする最新の暗号スイートも必要です。つまり、プライベートキーなどのシークレットが漏洩した場合でも、セッションキーは漏洩しません。

コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。標準 AWS KMS エンドポイントまたは FIPS AWS KMS エンドポイントを使用するには、クライアントが TLS 1.2 以降をサポートしている必要があります。利用可能な FIPS エンドポイントの詳細については、「[連邦情報処理規格 \(FIPS\) 140-3](#)」を参照してください。FIPS エンドポイントのリストについては、AWS KMS 「」の[AWS Key Management Service 「エンドポイントとクォータ」](#)を参照してください AWS 全般のリファレンス。

AWS KMS サービスホストと HSMs 間の通信は、認証された暗号化スキームで楕円曲線暗号化 (ECC) と高度な暗号化標準 (AES) を使用して保護されます。詳細については、AWS Key Management Service 「暗号化の詳細」の[「内部通信セキュリティ」](#)を参照してください。

インターネットトラフィックのプライバシー

AWS KMS は、AWS Management Console と一連の API オペレーションをサポートし、暗号化オペレーションで作成、管理 AWS KMS keys 、使用できるようにします。

AWS KMS は、プライベートネットワークから への 2 つのネットワーク接続オプションをサポートしています AWS。

- インターネット経由の IPsec VPN 接続
- [AWS Direct Connect](#): 内部ネットワークを標準のイーサネット光ファイバケーブル経由で AWS Direct Connect 口ケーションにリンクします。

すべての AWS KMS API コールは署名され、Transport Layer Security (TLS) を使用して送信する必要があります。コールには、[完全な転送秘密](#)をサポートする最新の暗号スイートも必要です。KMS キーのキーマテリアルを保存するハードウェアセキュリティモジュール (HSMs) へのトラフィックは、AWS 内部ネットワーク経由で既知の AWS KMS API ホストからのみ許可されます。

パブリックインターネット経由でトラフィックを送信せずに仮想プライベートクラウド (VPC) AWS KMS から に直接接続するには、 を搭載した VPC エンドポイントを使用します[AWS PrivateLink](#)。詳細については、「[VPC エンドポイント AWS KMS を介して に接続する](#)」を参照してください。

AWS KMS は、Transport Layer Security (TLS) ネットワーク暗号化プロトコルの[ハイブリッドポスト量子キー交換](#)オプションもサポートしています。このオプションは、AWS KMS API エンドポイントに接続するときに TLS で使用できます。

の ID とアクセスの管理 AWS Key Management Service

AWS Identity and Access Management (IAM) は、AWS リソースへのアクセスを安全に制御するのに役立ちます。管理者は、誰を認証 (サインイン) し、誰に AWS KMS リソースの使用を許可する (アクセス許可を付与する) かを制御します。詳細については、「[での IAM ポリシーの使用 AWS KMS](#)」を参照してください。

[キーポリシー](#)は、 で KMS キーへのアクセスを制御するための主要なメカニズムです AWS KMS。すべての KMS キーにはキーポリシーが必要です。[IAM ポリシー](#)と[権限](#)をキーポリシーとともに使用して、KMS キーへのアクセスを制御することもできます。詳細については、「[KMS キーのアクセスとアクセス許可](#)」を参照してください。

Amazon Virtual Private Cloud (Amazon VPC) を使用している場合は、AWS KMS を使用するインターフェイス [VPC エンドポイントを作成できますAWS PrivateLink](#)。VPC エンドポイントポリシーを使用して、AWS KMS エンドポイントにアクセスできるプリンシパル、実行できる API コール、アクセスできる KMS キーを決定することもできます。

トピック

- [AWS の 管理ポリシー AWS Key Management Service](#)
- [AWS KMSのサービスにリンクされたロールの使用](#)

AWS の 管理ポリシー AWS Key Management Service

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。AWS 管理ポリシーは、多くの一般的なユースケースにアクセス許可を付与するように設計されているため、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有の[カスタマー管理ポリシー](#)を定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使えるようになったときに、AWS 管理ポリシーを更新する可能性が高くなります。

詳細については「IAM ユーザーガイド」の「[AWS マネージドポリシー](#)」を参照してください。

AWS マネージドポリシー: AWSKeyManagementServicePowerUser

AWSKeyManagementServicePowerUser ポリシーを IAM アイデンティティにアタッチできます。

AWSKeyManagementServicePowerUser マネージドポリシーを使用して、アカウントの IAM プリンシパルにパワーユーザーの許可を付与できます。パワーユーザーは KMS キーを作成し、作成した KMS キーを使用および管理し、すべての KMS キーと IAM アイデンティティを表示できます。AWSKeyManagementServicePowerUser マネージドポリシーを持つプリンシパルは、キーポリシー、他の IAM ポリシー、許可など、他のソースから許可を取得することもできます。

AWSKeyManagementServicePowerUser は AWS マネージド IAM ポリシーです。AWS 管理ポリシーの詳細については、IAM ユーザーガイドの「[AWS 管理ポリシー](#)」を参照してください。

Note

KMS キーに固有のこのポリシー内の許可は (kms:TagResource および kms:GetKeyRotationStatus など)、その KMS キーのキーポリシーが、キーへのアクセスを制御するために IAM ポリシーを使用することを AWS アカウント に対して [明示的に許可している](#) 場合にのみ有効です。許可が KMS キーに固有のものであるかどうかを判断するには、[AWS KMS アクセス許可](#) を表示して、[Resources] (リソース) 列にある [KMS key] (KMS キー) の値を確認します。

このポリシーは、オペレーションを許可するキーポリシーを持つすべての KMS キーに対して、パワーユーザーの許可を付与します。クロスアカウントの許可の場合 (kms:DescribeKey および kms:ListGrants など)、これにより、信頼されていない AWS アカウントの KMS キーが含まれる可能性があります。詳細については、「[IAM ポリシーのベストプラクティス](#)」および「[他のアカウントのユーザーに KMS キーの使用を許可する](#)」を参照してください。許可が他のアカウントの KMS キーに対して有効かどうかを判断するには、[AWS KMS アクセス許可](#) を表示して、[Cross-account use] (クロスアカウントの使用) 列にある [Yes] (はい) の値を確認します。

プリンシパルがエラーなしで AWS KMS コンソールを表示できるようにするには、ポリシーに含まれていない [tag:GetResources](#) アクセス許可が必要です。

すAWSKeyManagementServicePowerUser。この許可は、別の IAM ポリシーで許可できます。

[AWSKeyManagementServicePowerUser](#) マネージド IAM ポリシーには、以下の許可が含まれます。

- プリンシパルが KMS キーを作成できるようにします。このプロセスにはキーポリシーの設定が含まれるため、パワーユーザーは自分や他者に、自分が作成した KMS キーを使用および管理するためのアクセス許可を付与することができます。
- プリンシパルは、すべての KMS キーの[エイリアス](#)と[タグ](#)を作成および削除できます。タグまたはエイリアスを変更すると、KMS キーを使用および管理するためのアクセス許可が、許可または拒否される場合があります。詳細については、「[の ABAC AWS KMS](#)」を参照してください。
- プリンシパルは、キー ARN、暗号化設定、キーポリシー、エイリアス、タグ、[ローテーション状態](#)など、すべての KMS キーに関する詳細情報を取得できます。
- プリンシパルが IAM ユーザー、グループ、ロールを一覧表示できるようにします。
- このポリシーでは、プリンシパルが自分で作成していない KMS キーを使用または管理することは許可できません。ただし、すべての KMS キーのエイリアスとタグを変更することはできるため、KMS キーを使用または管理する許可を許可または拒否できる可能性があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateAlias",
        "kms:CreateKey",
        "kms>DeleteAlias",
        "kms:Describe*",
        "kms:GenerateRandom",
        "kms:Get*",
        "kms:List*",
        "kms:TagResource",
        "kms:UntagResource",
        "iam:ListGroups",
        "iam:ListRoles",
```

```
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS マネージドポリシー:
AWSServiceRoleForKeyManagementServiceCustomKeyStores

IAM エンティティに AWSServiceRoleForKeyManagementServiceCustomKeyStores をアタッチすることはできません。このポリシーは、キーストアに関連付けられた AWS CloudHSM クラスターを表示し、カスタム AWS CloudHSM キーストアとその AWS CloudHSM クラスター間の接続をサポートするネットワークを作成するアクセス許可を付与 AWS KMS するサービスにリンクされたロールにアタッチされます。詳細については、「[AWS CloudHSM と Amazon EC2 リソースの管理 AWS KMS を承認する](#)」を参照してください。

AWS マネージドポリシー:
AWSServiceRoleForKeyManagementServiceMultiRegionKeys

IAM エンティティに AWSServiceRoleForKeyManagementServiceMultiRegionKeys をアタッチすることはできません。このポリシーは、マルチリージョンプライマリキーのキーマテリアルへの変更をレプリカキーに同期するアクセス許可を付与 AWS KMS するサービスにリンクされたロールにアタッチされます。詳細については、「[マルチリージョンキー AWS KMS の同期を許可する](#)」を参照してください。

AWS KMSAWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始 AWS KMS してからの の AWS 管理ポリシーの更新に関する詳細を表示します。このページへの変更に関する自動アラートについては、AWS KMS [ドキュメント履歴](#) ページの RSS フィードを購読してください。

変更	説明	日付
AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy – 既存のポリシーの更新	AWS KMS は、ポリシーバージョン v2 の マネージドポリシーにステートメント ID	2024 年 11 月 21 日

変更	説明	日付
	(Sid) フィールドを追加しました。	
AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy – 既存ポリシーの更新	AWS KMS は <code>ec2:DescribeVpcs</code> 、AWS CloudHSM クラスターを含む VPC の変更をモニタリングするための <code>ec2:DescribeNetworkAcls</code> 、および <code>Accessec2:DescribeNetworkInterfaces</code> 許可を追加し、が障害発生時に明確なエラーメッセージを提供 AWS KMS できるようにしました。	2023 年 11 月 10 日
AWS KMS が変更の追跡を開始しました	AWS KMS は、AWS 管理ポリシーの変更の追跡を開始しました。	2023 年 11 月 10 日

AWS KMSのサービスにリンクされたロールの使用

AWS Key Management Service は AWS Identity and Access Management (IAM) [サービスにリンクされたロール](#)を使用します。サービスにリンクされたロールは、直接リンクされた一意のタイプの IAM ロールです AWS KMS。サービスにリンクされたロールは によって定義 AWS KMS され、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれます。

サービスにリンクされたロールを使用すると、必要なアクセス許可を手動で追加する必要がなくなるため、 の設定 AWS KMS が簡単になります。 は、サービスにリンクされたロールのアクセス許可 AWS KMS を定義し、特に定義されている場合を除き、 のみがそのロールを引き受け AWS KMS することができます。定義される許可には、信頼ポリシーと許可ポリシーが含まれており、その許可ポリシーを他の IAM エンティティに添付することはできません。

サービスにリンクされたロールを削除するには、まずその関連リソースを削除します。これにより、AWS KMS リソースへのアクセス許可が誤って削除されないため、リソースが保護されます。

サービスにリンクされたロールをサポートする他のサービスについては、「[IAM と連携するAWS サービス](#)」を参照して、サービスにリンクされたロール列がはいになっているサービスを見つけてください。サービスにリンクされた役割に関するドキュメントをサービスで表示するには[はい] リンクを選択してください。

このトピックで説明されているサービスにリンクされたロールの更新の詳細については、「」を参照してください[AWS KMSAWS 管理ポリシーの更新](#)。

トピック

- [AWS CloudHSM と Amazon EC2 リソースの管理 AWS KMS を承認する](#)
- [マルチリージョンキー AWS KMS の同期を許可する](#)

AWS CloudHSM と Amazon EC2 リソースの管理 AWS KMS を承認する

AWS CloudHSM キーストアをサポートするために、には AWS CloudHSM クラスターに関する情報を取得するためのアクセス許可 AWS KMS が必要です。また、AWS CloudHSM キーストアを AWS CloudHSM クラスターに接続するネットワークインフラストラクチャを作成するアクセス許可も必要です。これらのアクセス許可を取得するには、で `AWSServiceRoleForKeyManagementServiceCustomKeyStores` サービスにリンクされたロール `AWS KMS` を作成します AWS アカウント。AWS CloudHSM キーストアを作成するユーザーには、サービスにリンクされたロールの作成を許可する `iam:CreateServiceLinkedRole` アクセス許可が必要です。

`AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy` 管理ポリシーの更新の詳細については、「」を参照してください[AWS KMSAWS 管理ポリシーの更新](#)。

トピック

- [AWS KMS サービスにリンクされたロールについて](#)
- [サービスにリンクされたロールの作成](#)
- [サービスにリンクされたロールの説明を編集する](#)
- [サービスにリンクされたロールを削除する](#)

AWS KMS サービスにリンクされたロールについて

[サービスにリンクされたロール](#)は、ユーザーに代わって他の AWS サービスを呼び出すアクセス許可を 1 つの AWS サービスに付与する IAM ロールです。複雑な IAM ポリシーを作成および維持しなく

ても、複数の統合 AWS サービスの機能を簡単に使用できるように設計されています。詳細については、「[AWS KMSのサービスにリンクされたロールの使用](#)」を参照してください。

AWS CloudHSM キーストアの場合、`AWSServiceRoleForKeyManagementServiceCustomKeyStores` サービスにリンクされたロールを `AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy` 管理ポリシーで AWS KMS 作成します。このポリシーはロールに以下のアクセス許可を与えます。

- [cloudhsm:Describe*](#) – カスタムキーストアにアタッチされている AWS CloudHSM クラスターの変更を検出します。
- [ec2:CreateSecurityGroup](#) – [AWS CloudHSM キーストアを接続して](#)、AWS KMS と AWS CloudHSM クラスター間のネットワークトラフィックフローを有効にするセキュリティグループを作成するときに使用されます。
- [ec2:AuthorizeSecurityGroupIngress](#) – [AWS CloudHSM キーストアを接続して](#)、から AWS CloudHSM クラスターを含む VPC AWS KMS へのネットワークアクセスを許可するときに使用されます。
- [ec2:CreateNetworkInterface](#) – [AWS CloudHSM キーストアを接続して](#)、AWS KMS と AWS CloudHSM クラスター間の通信に使用されるネットワークインターフェイスを作成するときに使用されます。
- [ec2:RevokeSecurityGroupEgress](#) – [AWS CloudHSM キーストアを接続して](#)、が AWS KMS 作成したセキュリティグループからすべてのアウトバウンドルールを削除するときに使用されます。
- [ec2:DeleteSecurityGroup](#) – [AWS CloudHSM キーストアを切断して](#)、キー AWS CloudHSM ストアの接続時に作成されたセキュリティグループを削除するときに使用されます。
- [ec2:DescribeSecurityGroups](#) – AWS CloudHSM クラスターを含む VPC で AWS KMS 作成されたセキュリティグループの変更をモニタリングし、が障害発生時に明確なエラーメッセージを提供 AWS KMS できるようにします。
- [ec2:DescribeVpcs](#) – AWS CloudHSM クラスターを含む VPC の変更をモニタリングし、が障害発生時に明確なエラーメッセージを提供 AWS KMS できるようにします。
- [ec2:DescribeNetworkAcls](#) – AWS CloudHSM クラスターを含む VPC のネットワーク ACLs の変更をモニタリングし、が障害発生時に明確なエラーメッセージを提供 AWS KMS できるようにします。
- [ec2:DescribeNetworkInterfaces](#) – AWS CloudHSM クラスターを含む VPC で AWS KMS 作成されたネットワークインターフェイスの変更をモニタリングし、AWS KMS が障害発生時に明確なエラーメッセージを提供できるようにします。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudhsm:Describe*",
        "ec2:CreateNetworkInterface",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateSecurityGroup",
        "ec2:DescribeSecurityGroups",
        "ec2:RevokeSecurityGroupEgress",
        "ec2>DeleteSecurityGroup",
        "ec2:DescribeVpcs",
        "ec2:DescribeNetworkAcls",
        "ec2:DescribeNetworkInterfaces"
      ],
      "Resource": "*"
    }
  ]
}
```

AWSServiceRoleForKeyManagementServiceCustomKeyStores サービスにリンクされたロールは、のみを信頼するため `cks.kms.amazonaws.com`、のみがこのサービスにリンクされたロールを引き受け AWS KMS することができます。このロールは、が AWS CloudHSM クラスターを表示し、AWS CloudHSM キーストアに関連する AWS CloudHSM クラスターに接続 AWS KMS するために必要なオペレーションに限定されます。追加のアクセス許可 AWS KMS は付与されません。たとえば、AWS KMS には、AWS CloudHSM クラスター、HSMs、またはバックアップを作成、管理、削除するアクセス許可はありません。

リージョン

AWS CloudHSM キーストア機能と同様

に、AWSServiceRoleForKeyManagementServiceCustomKeyStores ロールは、AWS KMS とが利用可能なすべての AWS リージョンでサポート AWS CloudHSM されています。各サービスがサポート AWS リージョンするのリストについては、の [AWS Key Management Service 「エンドポイントとクォータ」](#) および [AWS CloudHSM 「エンドポイントとクォータ」](#) を参照してください Amazon Web Services 全般のリファレンス。

AWS サービスにリンクされたロールの使用の詳細については、IAM ユーザーガイドの「[サービスにリンクされたロールの使用](#)」を参照してください。

サービスにリンクされたロールの作成

AWS KMS AWS CloudHSM は、キー AWS アカウント ストアを作成するとき、AWSServiceRoleForKeyManagementServiceCustomKeyStores サービスにリンクされたロールがまだ存在しない場合、自動的に AWSServiceRoleForKeyManagementServiceCustomKeyStores サービスにリンクされたロールを作成します。このサービスにリンクされたロールを直接作成または再作成することはできません。

サービスにリンクされたロールの説明を編集する

AWSServiceRoleForKeyManagementServiceServiceCustomKeyStores サービスリンクロールでは、ロール名またはポリシーステートメントを編集することはできませんが、ロールの説明を編集することはできます。手順については、「IAM ユーザーガイド」の「[サービスにリンクされたロールの編集](#)」を参照してください。

サービスにリンクされたロールを削除する

AWS KMS は、[すべての AWS CloudHSM キーストア](#)を削除した場合でも、AWSServiceRoleForKeyManagementServiceCustomKeyStores サービスにリンクされたロールを AWS アカウント から削除しません。現在、AWSServiceRoleForKeyManagementServiceCustomKeyStores サービスにリンクされたロールを削除する手順はありませんが、アクティブな AWS CloudHSM キーストアがない限り、このロールを引き受けたり、アクセス許可を使用した AWS KMS リすることはありません。

マルチリージョンキー AWS KMS の同期を許可する

[マルチリージョンキー](#)をサポートするには、マルチリージョンプライマリキーの[共有プロパティ](#)をレプリカキーと同期するアクセス許可 AWS KMS が必要です。これらのアクセス許可を取得するには、で AWSServiceRoleForKeyManagementServiceMultiRegionKeys サービスにリンクされたロール AWS KMS を作成します AWS アカウント。マルチリージョンキーを作成するユーザーには、サービスにリンクされたロールの作成を許可するiam:CreateServiceLinkedRoleアクセス許可が必要です。

AWS CloudTrail ログの共有プロパティの同期を記録する [SynchronizeMultiRegionKey](#) CloudTrail AWS KMS イベントを表示できます。

AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy 管理ポリシーの更新の詳細については、「」を参照してください[AWS KMSAWS 管理ポリシーの更新](#)。

トピック

- [マルチリージョンキーのサービスリンクロールについて](#)
- [サービスにリンクされたロールの作成](#)
- [サービスにリンクされたロールの説明を編集する](#)
- [サービスにリンクされたロールを削除する](#)

マルチリージョンキーのサービスリンクロールについて

[サービスにリンクされたロール](#)は、ユーザーに代わって他の AWS サービスを呼び出すアクセス許可を 1 つの AWS サービスに付与する IAM ロールです。複雑な IAM ポリシーを作成および維持しなくても、複数の統合 AWS サービスの機能を簡単に使用できるように設計されています。

マルチリージョンキーの場合、は、`AWSServiceRoleForKeyManagementServiceMultiRegionKeys` `AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy` サービスにリンクされたロール `AWS KMS` を作成します。このポリシーは、ロールに `kms:SynchronizeMultiRegionKey` アクセス許可を付与します。これにより、マルチリージョンキーの共有プロパティを同期できます。

`AWSServiceRoleForKeyManagementServiceMultiRegionKeys` サービスにリンクされたロールは のみを信頼するため `mrk.kms.amazonaws.com`、 のみがこのサービスにリンクされたロールを引き受け `AWS KMS` することができます。このロールは、マルチリージョン共有プロパティを同期 `AWS KMS` する必要があるオペレーションに限定されます。追加のアクセス許可 `AWS KMS` は付与されません。たとえば、`AWS KMS` には `KMS` キーを作成、レプリケート、削除するアクセス許可はありません。

`AWS` サービスにリンクされたロールの使用の詳細については、IAM ユーザーガイドの [「サービスにリンクされたロールの使用」](#) を参照してください。

JSON

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Sid" : "KMSSynchronizeMultiRegionKey",
      "Effect" : "Allow",
      "Action" : [
        "kms:SynchronizeMultiRegionKey"
      ],
    },
  ],
}
```

```
        "Resource" : "*"
    }
}
}
```

サービスにリンクされたロールの作成

AWS KMS ロールがまだ存在しない場合、マルチリージョンキーを作成する AWS アカウント と、`AWSServiceRoleForKeyManagementServiceMultiRegionKeys` サービスにリンクされたロールが自動的に作成されます。このサービスにリンクされたロールを直接作成または再作成することはできません。

サービスにリンクされたロールの説明を編集する

`AWSServiceRoleForKeyManagementServiceMultiRegionKeys` サービスリンクロールでは、ロール名またはポリシーステートメントを編集することはできませんが、ロールの説明を編集することができます。手順については、IAM ユーザーガイドの[サービスリンクロールの編集](#)を参照してください。

サービスにリンクされたロールを削除する

AWS KMS は、`AWSServiceRoleForKeyManagementServiceMultiRegionKeys` サービスにリンクされたロールを から削除せず AWS アカウント 、削除することはできません。ただし、AWS アカウント および リージョンにマルチリージョンキーがない限り、AWS KMS は `AWSServiceRoleForKeyManagementServiceMultiRegionKeys` ロールを引き受けたり、そのアクセス許可を使用したりしません。

でのログ記録とモニタリング AWS Key Management Service

モニタリングは、AWS KMSでの AWS KMS keys の可用性、ステータス、使用状況を理解する上で重要なパートです。モニタリングは、AWS ソリューションのセキュリティ、信頼性、可用性、パフォーマンスを維持するのに役立ちます。には、KMS キーをモニタリングするためのツールがいくつか AWS 用意されています。

AWS CloudTrail ログ

AWS KMS API オペレーションへのすべての呼び出しは、AWS CloudTrail ログにイベントとしてキャプチャされます。これらのログには、AWS KMS コンソールからのすべての API コールと、AWS KMS およびその他の AWS のサービスによって行われたコールが記録されます。ク

スアカウント API コール (別の AWS アカウントで KMS キーを使用する呼び出しなど) は、両方のアカウントの CloudTrail ログに記録されます。

トラブルシューティングまたは監査を行う際、ログを使用して KMS キーのライフサイクルを再構築できます。また、暗号化オペレーションにおける KMS キーの管理および使用を表示することもできます。詳細については、「[the section called “を使用したログ記録 AWS CloudTrail”](#)」を参照してください。

Amazon CloudWatch Logs

およびその他のソースからのログファイルをモニタリング、保存 AWS CloudTrail、およびアクセスします。詳細については、「[Amazon CloudWatch ユーザーガイド](#)」を参照してください。

CloudWatch には AWS KMS、KMS キーとそれらが保護するリソースに関する問題を防ぐのに役立つ情報が保存されます。詳細については、「[the section called “CloudWatch でキーをモニタリングする”](#)」を参照してください。

Amazon EventBridge

AWS KMS は、KMS キーが[ローテーションまたは削除されるか](#)、KMS キーに[インポートされたキーマテリアル](#)の有効期限が切れると、EventBridge イベントを生成します。[AWS KMS イベント \(API オペレーション\)](#)を検索し、1 つ以上のターゲット関数またはストリームにルーティングして、状態情報をキャプチャします。詳細については、[the section called “Amazon EventBridge を使用したキーのモニタリング”](#) および「[Amazon EventBridge ユーザーガイド](#)」を参照してください。

Amazon CloudWatch Metrics

CloudWatch メトリクスを使用して CloudWatch メトリクスは、から raw データを収集し、パフォーマンスメトリクス AWS KMS に処理します。データは 2 週間間隔で記録されるため、現在および過去の情報の傾向を表示できます。これにより、KMS キーがどのように使用され、それらの使用が時間の経過とともにどのように変化するかを理解できます。CloudWatch メトリクスを使用して KMS キーをモニタリングする方法については、[AWS KMS メトリクスとディメンション](#)を参照してください。

Amazon CloudWatch アラーム

指定した期間における単一のメトリックスの変化を監視します。次に、一定期間におけるしきい値に対するメトリックの値に基づいてアクションを実行します。例えば、暗号化オペレーションで削除がスケジュールされている KMS キーの使用を誰かが試みたときにトリガーされる CloudWatch アラームを作成できます。これは、KMS キーがまだ使用中であり、削除すべきでは

ないことを示します。詳細については、「[the section called “アラームの作成”](#)」を参照してください。

AWS Security Hub

を使用して、セキュリティ業界標準とベストプラクティスのコンプライアンスについて AWS KMS 使用状況をモニタリングできます AWS Security Hub。Security Hub は、セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようサポートします。詳細については、「AWS Security Hub ユーザーガイド」の「[AWS Key Management Service コントロール](#)」を参照してください。

のコンプライアンス検証 AWS Key Management Service

サードパーティーの監査者は、複数の コンプライアンスプログラム AWS Key Management Service の一環として のセキュリティと AWS コンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

トピック

- [コンプライアンスとセキュリティに関するドキュメント](#)
- [詳細](#)

コンプライアンスとセキュリティに関するドキュメント

以下のコンプライアンスおよびセキュリティドキュメントが対象です AWS KMS。表示するには、[AWS Artifact](#) を使用します。

- クラウドコンピューティングコンプライアンスコントロールカタログ (C5)
- ISO 27001:2013 適用宣言書 (SoA)
- ISO 27001:2013 認証
- ISO 27017:2015 適用宣言書 (SoA)
- ISO 27017:2015 認証
- ISO 27018:2015 適用宣言書 (SoA)
- ISO 27018:2014 認証
- ISO 9001:2015 認証
- PCI DSS Attestation of Compliance (AOC) と Responsibility Summary
- Service Organization Controls (SOC) 1 レポート

- Service Organization Controls (SOC) 2 レポート
- Service Organization Controls (SOC) 2 機密性に関するレポート
- フェドランプ高

の使用については AWS Artifact、[「Downloading Reports in AWS Artifact」](#) を参照してください。

詳細

を使用する際のお客様のコンプライアンス責任 AWS KMS は、お客様のデータの機密性、貴社のコンプライアンス目的、適用可能な法律および規制によって決まります。の使用 AWS KMS が公開されている標準に準拠していることを前提としている場合、は以下に役立つリソース AWS を提供します。

- [AWS コンプライアンスプログラムによる対象範囲内のサービス](#) – このページには、特定のコンプライアンスプログラムの対象となる AWS サービスが一覧表示されます。一般的な情報については、[「AWS コンプライアンスプログラム」](#) を参照してください。
- [セキュリティとコンプライアンスのクイックスタートガイド](#) – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイする手順を示します AWS。
- [AWS コンプライアンスリソース](#) – このワークブックとガイドのコレクションは、お客様の業界や場所に適用される場合があります。
- [AWS Config](#) – この AWS サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。
- [AWS Security Hub](#) – この AWS サービスは、内のセキュリティ状態を包括的に表示します AWS。Security Hub は、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールの一覧については、[Security Hub のコントロールリファレンス](#) を参照してください。

の耐障害性 AWS Key Management Service

AWS グローバルインフラストラクチャは、AWS リージョン およびアベイラビリティゾーンを中心に構築されています。は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された複数の物理的に分離および分離されたアベイラビリティゾーン AWS リージョン を提供します。アベイラビリティゾーンでは、アベイラビリティゾーン間で中断せずに、自動的に

フェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、およびスケーラビリティが優れています。

グローバル AWS インフラストラクチャに加えて、AWS KMS には、データの耐障害性とバックアップのニーズをサポートするのに役立ついくつかの機能が用意されています。AWS リージョンおよびアベイラビリティゾーンの詳細については、[AWS 「グローバルインフラストラクチャ」](#) を参照してください。

リージョンの隔離

AWS Key Management Service (AWS KMS) は、すべての で利用できる、自己維持型のリージョンサービスです AWS リージョン。のリージョン別に分離された設計 AWS KMS により、1 つの で可用性の問題が他の リージョンでの AWS KMS オペレーションに影響を与え AWS リージョン ないようことができます。AWS KMS は、計画されたダウンタイムをゼロにし、すべてのソフトウェア更新とスケーリングオペレーションをシームレスかつ目に見えない方法で実行できるように設計されています。

AWS KMS [サービスレベルアグリーメント](#) (SLA) には、すべての KMS APIs。このコミットメントを実現するために、AWS KMS は、API リクエストの実行に必要なすべてのデータと認可情報が、リクエストを受信するすべてのリージョンのホストで利用可能であることを確認します。

AWS KMS インフラストラクチャは、各リージョンの少なくとも 3 つのアベイラビリティゾーン (AZs) にレプリケートされます。複数のホスト障害が AWS KMS パフォーマンス AWS KMS に影響を与えないように、はリージョン内の任意の AZs からの顧客トラフィックに対応するように設計されています。

KMS キーのプロパティまたは許可に加えた変更は、リージョン内のすべてのホストにレプリケートされ、後続のリクエストがリージョン内の任意のホストで正しく処理されるようにします。KMS キーを使用した[暗号化オペレーション](#)のリクエストは、AWS KMS ハードウェアセキュリティモジュール (HSMs) のフリートに転送され、いずれのモジュールも KMS キーを使用してオペレーションを実行できます。

マルチテナント設計

のマルチテナント設計 AWS KMS により、99.999% の可用性 SLA を満たし、高いリクエストレートを維持しながら、キーとデータの機密性を維持できます。

暗号化オペレーションに指定した KMS キーが常に使用されるキーであることを保証するために、複数の整合性強制メカニズムがデプロイされます。

KMS キーのプレーンテキストキーマテリアルは、広範囲に保護されています。キーマテリアルは作成後すぐに HSM で暗号化され、暗号化されたキーマテリアルはセキュアで低レイテンシーのストレージに即座に移動されます。暗号化されたキーは、HSM 内で取得され、使用に間に合うように復号されます。プレーンテキストキーは、暗号化オペレーションを完了するのに必要な時間だけ HSM メモリに残ります。その後、HSM で再暗号化され、暗号化されたキーがストレージに返されます。プレーンテキストのキーマテリアルが HSM を離れることはありません。永続ストレージに書き込まれることもありません。

の耐障害性のベストプラクティス AWS KMS

AWS KMS リソースの耐障害性を最適化するには、次の戦略を検討してください。

- バックアップおよびディザスタリカバリ戦略をサポートするには、1 つの AWS リージョン で作成された KMS キーであり、指定したリージョンにのみレプリケートされる、マルチリージョンキーを検討します。マルチリージョンキーを使用すると、プレーンテキストを公開することなく AWS リージョン (同じパーティション内) 間で暗号化されたリソースを移動し、必要に応じて任意の送信先リージョンでリソースを復号できます。関連するマルチリージョンキーは、同じキーマテリアルとキー ID を共有するため、相互運用可能ですが、高解像度のアクセスコントロールのための独立したキーポリシーがあります。詳細については、[AWS KMSのマルチリージョンキー](#)を参照してください。
- などのマルチテナントサービスでキーを保護するには AWS KMS、[キーポリシー](#)や [IAM ポリシー](#)などのアクセスコントロールを必ず使用してください。さらに、を搭載した VPC インターフェイスエンドポイント AWS KMS を使用して、にリクエストを送信できます AWS PrivateLink。これを行うと、Amazon VPC と の間のすべての通信 AWS KMS は、VPC に制限された専用 AWS KMS エンドポイントを使用して AWS、ネットワーク内で完全に実施されます。[VPC エンドポイントポリシー](#)を使用して追加の認可レイヤーを作成することで、これらのリクエストをさらに保護できます。詳細については、「[VPC エンドポイント経由で AWS KMS に接続する](#)」を参照してください。

のインフラストラクチャセキュリティ AWS Key Management Service

マネージドサービスである AWS Key Management Service (AWS KMS) は、[「Amazon Web Services: セキュリティプロセスの概要」](#)で説明されている AWS グローバルネットワークセキュリティ手順で保護されています。

ネットワーク AWS KMS 経由で にアクセスするには、AWS KMS 「API [AWS Key Management Service リファレンス](#)」で説明されている API オペレーションを呼び出すことができます。AWS KMS は、すべてのリージョンで TLS 1.2 を必要とし、TLS 1.3 を推奨します。AWS KMS は、中国リージョンを除くすべてのリージョン AWS KMS のサービスエンドポイントのハイブリッドポスト量子 TLS もサポートしています。AWS KMS は、 の FIPS エンドポイントのハイブリッドポスト量子 TLS をサポートしていません AWS GovCloud (US)。 [標準の AWS KMS エンドポイント](#)または [AWS KMS FIPS エンドポイント](#)を使用するには、クライアントが TLS 1.2 以降をサポートする必要があります。また、Ephemeral Diffie-Hellman (DHE) や Elliptic Curve Ephemeral Diffie-Hellman (ECDHE) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートもクライアントでサポートされている必要があります。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または[AWS Security Token Service](#) (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

これらの API オペレーションは任意のネットワークの場所から呼び出すことができますが、 は、ソース IP アドレス、VPC、VPC エンドポイントに基づいて KMS キーへのアクセスを制御できるグローバルポリシー条件 AWS KMS をサポートしています。これらの条件キーは、キーポリシーと IAM ポリシーで使用できます。ただし、これらの条件により、 ユーザーに代わって KMS キーを使用すること AWS を防ぐことができます。詳細については、「[AWS グローバル条件キー](#)」を参照してください。

例えば、次のキーポリシーステートメントでは、ソース IP アドレスがポリシーで指定された IP アドレスの 1 つでない限り、指定された[暗号化オペレーション](#) AWS KMS key にこれを使用できるKMSTestRoleロールを引き受けることができるユーザーに許可します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS":
      "arn:aws:iam::111122223333:role/KMSTestRole"},
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncrypt*",
```

```
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*",
  "Condition": {
    "NotIpAddress": {
      "aws:SourceIp": [
        "192.0.2.0/24",
        "203.0.113.0/24"
      ]
    }
  }
}
```

物理ホストの分離

が AWS KMS 使用する物理インフラストラクチャのセキュリティは、「Amazon Web Services: セキュリティプロセスの概要」の「物理的および環境的なセキュリティ」セクションで説明されているコントロールの対象となります。 https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf 詳細については、前のセクションにリストされたコンプライアンスレポートとサードパーティーの監査結果を参照してください。

AWS KMS は、物理攻撃に耐えるための特定のコントロールで設計された専用の強化ハードウェアセキュリティモジュール (HSMs) でサポートされています。HSM は、ハイパーバイザーなどの仮想化レイヤーを持たない物理デバイスで、物理デバイスを複数の論理テナント間で共有します。AWS KMS keys のキーマテリアルは、KMS キーの使用中にのみ、HSM の揮発性メモリにのみ保存されます。このメモリは、HSM が意図したシャットダウンやリセットなど、動作状態から移行すると消去されます。AWS KMS HSMs [AWS Key Management Service 「暗号化の詳細」](#) を参照してください。

クォータ

すべてのユーザーに対して AWS KMS 応答性とパフォーマンスを実現するために、はリソースクォータとリクエストクォータの 2 種類のクォータ AWS KMS を適用します。各クォータは、各 AWS アカウントのリージョンごとに個別に計算されます。

オンデマンドローテーションリソース AWS KMS クォータと [AWS CloudHSM キーストアリクエストクォータ](#)を除き、すべてのクォータは調整可能です。[???クォータの引き上げをリクエストするには](#)、「Service Quotas ユーザーガイド」の「[クォータ引き上げリクエスト](#)」を参照してください。クォータの引き下げをリクエストしたり、Service Quotas にリストされていないクォータを変更したり、の Service Quotas AWS リージョン が AWS KMS 利用できない のクォータを変更したりするには、[AWS サポート センター](#)にアクセスしてケースを作成してください。

トピック

- [リソースクォータ](#)
- [クォータのリクエスト](#)
- [AWS KMS リクエストのスロットリング](#)

リソースクォータ

AWS KMS は、すべてのお客様に迅速かつ回復力のあるサービスを提供できるように、リソースクォータを確立します。一部のリソースクォータは、作成したリソースにのみ適用されますが、AWS サービスが作成するリソースには適用されません。[AWS 所有のキー](#) など、使用するリソースで AWS アカウントに含まれていないものは、これらのクォータにはカウントされません。

リソースの上限を超えた場合、そのタイプの追加リソースの作成をリクエストすると、LimitExceededException エラーメッセージが生成されます。

オンデマンドローテーション AWS KMS リソースクォータを除き、すべてのリソースクォータは調整可能です。[???クォータの引き上げをリクエストするには](#)、「Service Quotas ユーザーガイド」の「[クォータ引き上げリクエスト](#)」を参照してください。クォータの削減をリクエストする、Service Quotas にリストされていないクォータを変更する、または Service Quotas for AWS リージョン が AWS KMS 利用できない のクォータを変更するには、[AWS サポート センター](#)にアクセスしてケースを作成してください。

次の表は、各 およびリージョンの AWS KMS リソースクォータの一覧 AWS アカウント と説明です。

クォータ名	デフォルト値	適用先	調整可能
AWS KMS keys	100,000	カスタマーマネージドキー	はい
KMS キーごとのエイリアス	50	カスタマー作成のエイリアス	はい
KMS キーごとのグラント	50,000	カスタマーマネージドキー	はい
カスタムキーストアのリソースクォータ	10	AWS アカウント および リージョン	はい
オンデマンドローテーション	10	カスタマーマネージドキー	いいえ

リソースクォータに加えて、はリクエストクォータ AWS KMS を使用してサービスの応答性を確保します。詳細については、「[the section called “クォータのリクエスト”](#)」を参照してください。

AWS KMS keys: 100,000

AWS アカウントの各リージョンで、最大 100,000 個の[カスタマーマネージドキー](#)を持つことができます。このクォータは、[キー仕様](#)または[キーステータス](#)に関係なく、すべての AWS リージョンですべてのカスタマー管理キーに適用されます。各 KMS キーは 1 つのリソースと見なされます。[AWS マネージドキー](#) および [AWS 所有のキー](#) はこのクォータにはカウントされません。

KMS キーごとのエイリアス: 50

最大 50 個の[エイリアス](#)を各[カスタマーマネージドキー](#)に関連付けることができます。が AWS 関連付けるエイリアス[AWS マネージドキー](#)は、このクォータにはカウントされません。エイリアスを[作成](#)または[更新](#)するときに、このクォータが発生することがあります。

Note

[kms:ResourceAliases](#) 条件は、KMS キーがこのクォータに適合している場合にのみ有効です。KMS キーがこのクォータを超えると、KMS キーを `kms:ResourceAliases` 条件で使

用するよう認可されたプリンシパルは、KMS キーへのアクセスを拒否されます。詳細については、「[エイリアスクォータによりアクセスが拒否された](#)」を参照してください。

KMS キークォータあたりのエイリアスは、の各リージョンのエイリアスの合計数を制限するリージョンクォータあたりのエイリアスを置き換えます AWS アカウント。AWS KMS はリージョンクォータあたりのエイリアスを削除しました。

KMS キーあたりのグラント: 50,000

各[カスターマネージドキー](#)は、[AWS KMSと統合されているAWS サービス](#)によって作成されるグラントを含めて、最大 50,000 個の[グラント](#)を持つことができます。このクォータは、[AWS マネージドキー](#) または [AWS 所有のキー](#) には適用されません。

このクォータの効果の 1 つは、同じ KMS キーを同時に使用する 50,000 を超える許可されたオペレーションを実行できないことです。このクォータ到達後は、アクティブなグラントが廃止または取り消しになった場合にのみ、KMS キーで新しいグラントを作成できます。

例えば、Amazon Elastic Block Store (Amazon EBS) ボリュームを Amazon Elastic Compute Cloud (Amazon EC2) インスタンスにアタッチすると、ボリュームは復号化され、読めるようになります。データを復号する許可を得るために、Amazon EBS は各ボリュームに対してグラントを作成します。したがって、すべての Amazon EBS ボリュームが同じ KMS キーを使用する場合、一度に 50,000 を超えるボリュームをアタッチすることはできません。

カスタムキーストアのリソースクォータ: 10

AWS アカウント およびリージョンごとに最大 10 個の[カスタムキーストア](#)を作成できます。さらに作成しようとする、[CreateCustomKeyStore](#) オペレーションは失敗します。

このクォータは、接続状態に関わりなく、すべての[AWS CloudHSM キーストア](#)と[外部キーストア](#)を含む、各アカウントとリージョンのカスタムキーストアの合計数に適用されます。

オンデマンドローテーション: 10

[オンデマンドキーローテーション](#)は、1 つの KMS キーごとに最大 10 回実行できます。これより多くのオンデマンドローテーションを実行しようとする、[RotateKeyOnDemand](#) オペレーションは失敗します。

これは調整可能なクォータではありません。Service Quotas を使用するか、でケースを作成することで増やすことはできません AWS サポート。オンデマンドローテーションクォータに到達するのを避けるため、可能な限り[自動キーローテーション](#)を使用することをお勧めします。

クォータのリクエスト

AWS KMS は、1 秒ごとにリクエストされた API オペレーションの数のクォータを確立します。リクエストクォータは、API オペレーション AWS リージョン、および KMS キータイプなどのその他の要因によって異なります。API リクエストのクォータを超えると、AWS KMS [はリクエストをスロットリングします](#)。

キーストア AWS KMS のリクエストクォータを除き、すべてのリクエストクォータは調整可能です。[AWS CloudHSM](#) クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「[クォータ引き上げリクエスト](#)」を参照してください。クォータの削減をリクエストする、Service Quotas にリストされていないクォータを変更する、または Service Quotas for AWS リージョン が AWS KMS 利用できない のクォータを変更するには、[AWS サポート Center](#) にアクセスしてケースを作成してください。

[GenerateDataKey](#) オペレーションのリクエストクォータを超えている場合は、AWS Encryption SDKの[データキーキャッシュ機能](#)の使用を検討してください。データキーを再利用すると、へのリクエストの頻度が低下する可能性があります AWS KMS。

リクエストクォータに加えて、はリソースクォータ AWS KMS を使用してすべてのユーザーの容量を確保します。詳細については、「[リソースクォータ](#)」を参照してください。

リクエストレートのトレンドを表示するには、[Service Quotas コンソール](#)を使用してください。リクエストレートがクォータ値の一定の割合に達したときに警告する [Amazon CloudWatch](#) アラームを作成することもできます。詳細については、AWS セキュリティブログの[Service Quotas と Amazon CloudWatch を使用して AWS KMS API リクエストレートを管理する](#)」を参照してください。

トピック

- [各 AWS KMS API オペレーションのリクエストクォータ](#)
- [リクエストクォータの適用](#)
- [暗号化オペレーションの共有クォータ](#)
- [ユーザーに代わって API が実行するリクエスト](#)
- [クロスアカウントリクエスト](#)
- [カスタムキーストアのリクエストクォータ](#)

各 AWS KMS API オペレーションのリクエストクォータ

この表は、各 AWS KMS リクエストクォータの [Service Quotas](#) クォータコードとデフォルト値を示しています。キーストア AWS KMS のリクエストクォータを除き、すべてのリクエストクォータは調整可能です。 [AWS CloudHSM](#)

Note

この表のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

クォータ名	デフォルト値 (1 秒あたりのリクエスト)
Cryptographic operations (symmetric) request rate 適用先: - Decrypt - Encrypt - GenerateDataKey - GenerateDataKeyWithoutPlaintext - GenerateMac - GenerateRandom - ReEncrypt - VerifyMac	これらの共有クォータは、AWS リージョン およびリクエストで使用される KMS キーのタイプによって異なります。各クォータは個別に計算されます。 10,000 (共有) - 以下のリージョンでは 20,000 (共有): - 米国東部 (オハイオ)、us-east-2 - アジアパシフィック (シンガポール)、ap-southeast-1 - アジアパシフィック (シドニー)、ap-southeast-2 - アジアパシフィック (東京)、ap-northeast-1 - ヨーロッパ (フランクフルト)、eu-central-1 - ヨーロッパ (ロンドン)、eu-west-2 - 以下のリージョンでは 100,000 (共有): - 米国東部 (バージニア北部)、us-east-1 - 米国西部 (オレゴン)、us-west-2 - ヨーロッパ (アイルランド)、eu-west-1

クォータ名	デフォルト値 (1 秒あたりのリクエスト)
Cryptographic operations (RSA) request rate 適用先: • Decrypt • Encrypt • ReEncrypt • Sign • Verify	RSA KMS キーの場合は 1,000 (共有)
Cryptographic operations (ML-DSA) request rate 適用先: • Sign • Verify	ML-DSA KMS キーの場合は 1,000 (共有)
Cryptographic operations (ECC and SM2) request rate 適用先: • Decrypt — SM2 (中国リージョンのみ) KMS キーでのみサポート • DeriveSharedSecret • Encrypt — SM2 (中国リージョンのみ) KMS キーでのみサポート • ReEncrypt — SM2 (中国リージョンのみ) KMS キーでのみサポート • Sign • Verify	楕円曲線 (ECC) KMS キーおよび SM2 (中国リージョンのみ) KMS キーの場合は 1,000 (共有)

クォータ名	デフォルト値 (1 秒あたりのリクエスト)
Custom key store request quotas 適用先: • Decrypt • DeriveSharedSecret • Encrypt • GenerateDataKey • GenerateDataKeyWithoutPlainText • GenerateRandom • ReEncrypt	カスタムキーストアのリクエストクォータ は、カスタムキーストアごとに個別に計算されます。 • AWS CloudHSM キーストアごとに 1,800 (共有) • 外部キーストアごとに 1,800 (共有)
CancelKeyDeletion request rate	5
ConnectCustomKeyStore request rate	5
CreateAlias request rate	5
CreateCustomKeyStore request rate	5
CreateGrant request rate	50
CreateKey request rate	5
DeleteAlias request rate	15
DeleteCustomKeyStore request rate	5
DeleteImportedKeyMaterial request rate	15
DescribeCustomKeyStores request rate	5
DescribeKey request rate	2000

クォータ名	デフォルト値 (1 秒あたりのリクエスト)
DisableKey request rate	5
DisableKeyRotation request rate	5
DisconnectCustomKeyStore request rate	5
EnableKey request rate	5
EnableKeyRotation request rate	15
GenerateDataKeyPair (ECC_NIST_P256) request rate 適用先: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100
GenerateDataKeyPair (ECC_NIST_P384) request rate 適用先: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100
GenerateDataKeyPair (ECC_NIST_P521) request rate 適用先: • GenerateDataKeyPair • GenerateDataKeyPairWithoutPlaintext	100

クォータ名	デフォルト値 (1 秒あたりのリクエスト)
<code>GenerateDataKeyPair (ECC_SECG_P256K1) request rate</code> 適用先: • <code>GenerateDataKeyPair</code> • <code>GenerateDataKeyPairWithoutPlaintext</code>	100
<code>GenerateDataKeyPair (RSA_2048) request rate</code> 適用先: • <code>GenerateDataKeyPair</code> • <code>GenerateDataKeyPairWithoutPlaintext</code>	1
<code>GenerateDataKeyPair (RSA_3072) request rate</code> 適用先: • <code>GenerateDataKeyPair</code> • <code>GenerateDataKeyPairWithoutPlaintext</code>	0.5 (2 秒に 1 回)
<code>GenerateDataKeyPair (RSA_4096) request rate</code> 適用先: • <code>GenerateDataKeyPair</code> • <code>GenerateDataKeyPairWithoutPlaintext</code>	0.1 (10 秒に 1 回)

クォータ名	デフォルト値 (1 秒あたりのリクエスト)
GenerateDataKeyPair (SM2 – China Regions only) request rate	25
適用先:	
- GenerateDataKeyPair - GenerateDataKeyPairWithoutPlaintext	
GetKeyPolicy request rate	1,000
GetKeyRotationStatus request rate	1,000
GetParametersForImport request rate	0.25 (4 秒に 1 回)
GetPublicKey request rate	2000
ImportKeyMaterial request rate	15
ListAliases request rate	500
ListGrants request rate	100
ListKeyPolicies request rate	100
ListKeys request rate	500
ListKeyRotations request rate	100
ListResourceTags request rate	2000
ListRetirableGrants request rate	100
PutKeyPolicy request rate	15

クォータ名	デフォルト値 (1 秒あたりのリクエスト)
ReplicateKey request rate ReplicateKey オペレーションは、プライマリキーのリージョンでは 1 つ ReplicateKey リクエスト、レプリカのリージョンでは 2 つの CreateKey リクエストとしてカウントされます。以下のうち 1 つの CreateKey リクエストは、キー作成前に潜在的な問題を検出するためのドライランです。	5
RetireGrant request rate	50
RevokeGrant request rate	50
RotateKeyOnDemand request rate	5
ScheduleKeyDeletion request rate	15
TagResource request rate	10
UntagResource request rate	5
UpdateAlias request rate	5
UpdateCustomKeyStore request rate	5
UpdateKeyDescription request rate	5
UpdatePrimaryRegion request rate UpdatePrimaryRegion オペレーションは、2 つの影響を受けるリージョンごとに、1 つのリクエストとしてカウントされる、2 つの UpdatePrimaryRegion リクエストです。	5

リクエストクォータの適用

リクエストクォータを確認するときは、次の点に注意してください。

- リクエストクォータは、[カスタマーマネージドキー](#)と [AWS マネージドキー](#) の両方に適用されます。の使用は、アカウントのリソースを保護するために使用されている場合でも AWS アカウント、のリクエストクォータにはカウント [AWS 所有のキー](#) されません。
- リクエストクォータは、FIPS エンドポイントおよび非 FIPS エンドポイントに送信されるリクエストに適用されます。AWS KMS サービスエンドポイントのリストについては、「」の [AWS Key Management Service 「エンドポイントとクォータ」](#) を参照してください AWS 全般のリファレンス。
- スロットリングは、リージョン内のすべてのタイプの KMS キーに対するすべてのリクエストに基づいています。この合計には、ユーザーに代わって のサービスからのリクエストを含む AWS アカウント、 のすべてのプリンシパルからの AWS リクエストが含まれます。
- 各要求クォータは個別に計算されます。例えば、[CreateKey](#) オペレーションのリクエストは、[CreateAlias](#) オペレーションのリクエストクォータには影響しません。CreateAlias リクエストが調整されていても、CreateKey リクエストは正常に完了できます。
- 暗号化オペレーションはクォータを共有しますが、共有クォータは他のオペレーションのクォータとは無関係に計算されます。例えば、[Encrypt](#) および [Decrypt](#) オペレーションの呼び出しはリクエストクォータを共有しますが、そのクォータは [EnableKey](#) などの管理オペレーションのクォータとは無関係です。例えば、欧州 (ロンドン) リージョンでは、対称 KMS キーに対して 10,000 オペレーションの暗号化オペレーションに加えて、スロットルなしで 1 秒あたり 5 回の EnableKey オペレーションを実行できます。

暗号化オペレーションの共有クォータ

AWS KMS [暗号化オペレーション](#) はリクエストクォータを共有します。KMS キーでサポートされている暗号化オペレーションの任意の組み合わせをリクエストできます。これにより、暗号化オペレーションの合計数がそのタイプの KMS キーのリクエストクォータを超過しません。例外は、別個のクォータを共有する [GenerateDataKeyPair](#) と [GenerateDataKeyPairWithoutPlaintext](#) です。

異なるタイプの KMS キーのクォータは、個別に計算されます。各クォータは、1 秒間隔で指定されたキータイプを持つ AWS アカウント および リージョンで、これらのオペレーションに対するすべてのリクエストに適用されます。

- 暗号化オペレーション (対称) リクエストの頻度は、アカウントとリージョンで対称 KMS キーを使用する暗号化オペレーションの共有リクエストクォータです。このクォータは、対称暗号化キーと、同じく対称である HMAC キーでの暗号化オペレーションに適用されます。

たとえば、1 秒あたり 10,000 リクエストの共有クォータ AWS リージョン を持つ [で対称 KMS キー](#)を使用しているとします。1 秒あたり 7,000 件の [GenerateDataKey](#) リクエストと 1 秒あたり 2,000 件の [Decrypt](#) リクエストを行う場合、AWS KMS はリクエストを調整しません。ただし、毎秒 9,500 件の [GenerateDataKey](#) リクエストと 1,000 件の [Encrypt](#) リクエストを行うと、共有クォータを超えているため、AWS KMS はリクエストを制限します。

[カスタムキーストア](#)内の[対称暗号化 KMS キー](#)に対する暗号化オペレーションは、アカウントの暗号化オペレーション (対称) リクエストレートおよびカスタムキーストアの[カスタムキーストアの リクエストクォータ](#)の両方にカウントされます。

- 暗号化オペレーション (RSA) リクエストの頻度は、[RSA 非対称 KMS キー](#)を使用する暗号化オペレーションの共有リクエストクォータです。

例えば、1 秒あたり 1,000 オペレーションのリクエストクォータでは、暗号化および復号が可能な RSA KMS キーを使用して、400 件の [Encrypt](#) リクエストと 200 件の [Decrypt](#) リクエストを実行できます。加えて、署名と検証が可能な RSA KMS キーを使用して、250 件の [Sign](#) リクエストと 150 件の [Verify](#) リクエストを実行できます。

- 暗号化オペレーション (ECC) リクエストレートは、[楕円曲線 \(ECC\) 非対称 KMS キー](#)および [SM 非対称 KMS キー](#)を使用する暗号化オペレーションの共有リクエストクォータです。

例えば、1 秒あたり 1,000 オペレーションのリクエストクォータでは、署名と検証が可能な ECC KMS キーを使用して、400 件の [Sign](#) リクエストと 200 件の [Verify](#) リクエストを実行できます。加えて、署名と検証が可能な SM2 KMS キーを使用して、250 件の [Sign](#) リクエストと 150 件の [Verify](#) リクエストを実行できます。

- カスタムキーストアのリクエストクォータは、カスタムキーストアの KMS キーに対する暗号化オペレーションの共有リクエストクォータです。このクォータは、各カスタムキーストアで個別に計算されます。

[カスタムキーストア](#)内の[対称暗号化 KMS キー](#)に対する暗号化オペレーションは、アカウントの暗号化オペレーション (対称) リクエストレートおよびカスタムキーストアの[カスタムキーストアの リクエストクォータ](#)の両方にカウントされます。

異なるキータイプのクォータも個別に計算されます。例えば、アジアパシフィック (シンガポール) リージョンでは、対称および非対称の KMS キーの両方を使用する場合、対称 KMS キー (HMAC

キーを含む) を使用した 1 秒あたり最大 10,000 件の呼び出しに加えて、RSA 非対称 KMS キーを使用した 1 秒あたり最大 500 件の追加呼び出しと、ECC ベースの KMS キーを使用した 1 秒あたり最大 300 件の追加リクエストを実行できます。

ユーザーに代わって API が実行するリクエスト

API リクエストは、直接行うか、AWS KMS ユーザーに代わって に API リクエストを行う統合 AWS サービスを使用して行うことができます。クォータはどちらの種類のリクエストにも適用されます。

たとえば、KMS キー (SSE-KMS) によるサーバー側の暗号化を使用して Amazon S3 にデータを保存できます。SSE-KMS で暗号化された S3 オブジェクトをアップロードまたはダウンロードするたびに、Amazon S3 は AWS KMS ユーザーに代わって に GenerateDataKey (アップロードの場合) または Decrypt (ダウンロードの場合) リクエストを行います。これらのリクエストはクォータにカウントされるため、SSE-KMS で暗号化された S3 オブジェクトの 1 秒あたりのアップロードまたはダウンロードの合計数が 5,500 (または に応じて 10,000 または 50,000 AWS リージョン) を超える場合、 はリクエスト AWS KMS を調整します。

クロスアカウントリクエスト

あるアプリケーションが別のアカウントが所有する KMS キー AWS アカウント を使用する場合、クロスアカウントリクエストと呼ばれます。クロスアカウントリクエストでは、AWS KMS は、KMS キーを所有するアカウントではなく、リクエストを行うアカウントを調整します。例えば、アカウント A のアプリケーションがアカウント B の KMS キーを使用する場合、KMS キーの使用はアカウント A のクォータにのみ適用されます。

カスタムキーストアのリクエストクォータ

AWS KMS は、[カスタムキーストア](#)の KMS キーに対する[暗号化オペレーション](#)のリクエストクォータを維持します。これらのリクエストクォータは、カスタムキーストアごとに個別に計算されます。

カスタムキーストアのリクエストクォータ	各カスタムキーストアのデフォルト値 (リクエスト数/秒)	調整可能
AWS CloudHSM キーストア リクエストクォータ	1800	いいえ
外部キーストア のリクエストクォータ	1800	はい

Note

AWS KMS [カスタムキーストアリクエストクォータ](#)は Service Quotas コンソールに表示されません。Service Quotas API オペレーションを使用して、これらのクォータを表示または管理することはできません。外部キーストアのリクエストクォータへの変更をリクエストするには、[AWS サポート センター](#)にアクセスしてケースを作成します。

AWS CloudHSM キーストアに関連付けられた AWS CloudHSM クラスターが、カスタムキーストアに関連しないコマンドなど、多数のコマンドを処理している場合、は AWS KMS 予想よりも低いレート `ThrottlingException` で取得される可能性があります。lower-than-expected この場合、リクエストレートを に下げるか AWS KMS、無関係な負荷を減らすか、AWS CloudHSM キーストアに専用 AWS CloudHSM クラスターを使用します。

AWS KMS は、[ExternalKeyStoreThrottle](#) CloudWatch メトリクスで外部キーストア リクエストのスロットリングを報告します。このメトリクスを使用して、スロットリングパターンを表示したり、アラームを作成したり、外部キーストアのリクエストクォータを調整したりできます。

カスタムキーストアの KMS キーに対する [暗号化オペレーション](#) のリクエストは、2 クォータにカウントされます。

- 暗号化オペレーション (対称) のリクエストレートクォータ (アカウントあたり)

カスタムキーストアの KMS キーに対する暗号化オペレーションのリクエストは、各 AWS アカウント およびリージョンの Cryptographic operations (symmetric) request rate クォータにカウントされます。たとえば、米国東部 (バージニア北部) (us-east-1) では、カスタムキーストアで KMS キーを使用するリクエストを含め、対称暗号化 KMS キーに対して 1 秒あたり最大 100,000 リクエストを持つ AWS アカウント ことができます。

- カスタムキーストアのリクエストクォータ (カスタムキーストアあたり)

カスタムキーストアの KMS キーに対する暗号化オペレーションのリクエストも、1 秒あたり 1,800 オペレーションの Custom key store request quota にカウントされます。これらのクォータは、カスタムキーストアごとに個別に計算されます。これには、カスタムキーストアで KMS キー AWS アカウント を使用する複数の からのリクエストが含まれる場合があります。

たとえば、米国東部 (バージニア北部) (us-east-1) リージョンのカスタムキーストア (タイプを問わず) の KMS キーに対する [Encrypt](#) オペレーションは、そのアカウントとリージョンの Cryptographic operations (symmetric) request rate アカウントレベルのクォータ

(100,000 リクエスト/秒) と、カスタムキーストアの Custom key store request quota (1,800 リクエスト/秒) にカウントされます。ただし、カスタムキーストア内の KMS キーに対する管理操作 ([PutKeyPolicy](#) など) のリクエストは、アカウントレベルのクォータ (1 秒あたり 15 リクエスト) によりのみ適用されます。

AWS KMS リクエストのスロットリング

は、すべての顧客からの API リクエストに迅速かつ信頼性の高いレスポンスを提供 AWS KMS できるように、特定の境界を超える API リクエストを調整します。

スロットリングは、 がそれ以外の場合に有効なリクエスト AWS KMS を拒否し、次のような `ThrottlingException` エラーを返す場合に発生します。

```
You have exceeded the rate at which you may call KMS. Reduce the frequency of your calls.
(Service: AWSKMS; Status Code: 400; Error Code: ThrottlingException; Request ID: <ID>
```

AWS KMS は、次の条件のリクエストを調整します。

- 1 秒あたりのリクエストのレートが、アカウントとリージョンの AWS KMS [リクエストクォータ](#) を超えています。

例えば、アカウントのユーザーが 1 秒間に 1000 件の `DescribeKey` リクエストを送信した場合、はその 1 秒間に後続のすべての `DescribeKey` リクエスト AWS KMS をスロットリングします。

スロットリングに対応するには、[バックオフと再試行戦略](#) を使用します。この戦略は、一部の AWS SDKs。

- 同じ KMS キーステータスを変更するための、リクエストのバーストレートまたは持続的な高いレート。多くの場合、この条件は「ホットキー」と呼ばれます。

例えば、アカウント内のアプリケーションが、同じ KMS キーに対する `EnableKey` および `DisableKey` リクエストの永続的なボリュームを送信した場合、 はリクエストを AWS KMS スロットリングします。このスロットリングは、リクエストが `EnableKey` および `DisableKey` オペレーションの 1 秒あたりのリクエスト制限を超えていない場合にも発生します。

スロットリングに対応するには、アプリケーションロジックを調整して必要なリクエストのみを作成するか、複数の関数のリクエストを統合します。

- [AWS CloudHSM キーストア](#) に関連付けられたクラスターが、キーストアに関連しないコマンドを含む多数のコマンドを処理している場合 AWS CloudHSM 、 AWS CloudHSM キーストア内の

KMS キーに対するオペレーションのリクエストはlower-than-expectedレートでスロットリングされる可能性があります AWS CloudHSM。

(AWS CloudHSM クラスターで使用可能な PKCS #11 セッションがない場合、 は AWS CloudHSM キーストア内の KMS キーに対するオペレーションのリクエストをスロットリング AWS KMS しなくなりました。代わりに ガスローKMSInternalExceptionされ、リクエストを再試行することをお勧めします)。

リクエストレートのトレンドを表示するには、[Service Quotas コンソール](#)を使用してください。リクエストレートがクォータ値の一定の割合に達したときに警告する [Amazon CloudWatch](#) アラームを作成することもできます。詳細については、AWS セキュリティブログの[Service Quotas と Amazon CloudWatch を使用して AWS KMS API リクエストレートを管理する](#)」を参照してください。

オンデマンドローテーションリソース AWS KMS クォータと[AWS CloudHSM キーストアリクエストクォータ](#)を除き、[すべてのクォータ](#)は調整可能です。[???クォータ](#)の引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「[クォータ引き上げリクエスト](#)」を参照してください。クォータの引き下げをリクエストしたり、Service Quotas にリストされていないクォータを変更したり、 の Service Quotas AWS リージョン が AWS KMS 利用できない のクォータを変更したりするには、[AWS サポート センター](#)にアクセスしてケースを作成してください。

 Note

AWS KMS [カスタムキーストアのリクエストクォータ](#)は Service Quotas コンソールに表示されません。Service Quotas API オペレーションを使用して、これらのクォータを表示または管理することはできません。外部キーストアのリクエストクォータへの変更をリクエストするには、[AWS サポート センター](#)にアクセスしてケースを作成します。

AWS SDKs AWS KMS を使用するためのコード例

次のコード例は、AWS Software Development Kit (SDK) AWS KMS で 使用方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

「シナリオ」は、1つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

こんにちは AWS Key Management Serviceは

次のコード例は、AWS Key Management Serviceの使用を開始する方法を示しています。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * 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
*/
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
        AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
        which is
         * based on a reactive stream. This allows asynchronous retrieval of
        paginated
         * results as they become available. By subscribing to the stream, we can
        process
         * each page of results as they are emitted.
        */
        ListKeysPublisher keysPublisher =
        kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
        Id is: " + key.keyId()))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " +
        exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });
    }
}
```



```
        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListKeys](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get
the keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or
to break up the result into pages.

$keys = [];
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);
foreach($keysPaginator as $page){
    foreach($page['Keys'] as $index => $key){
        echo "The $index index Key's ID is: {$key['KeyId']}\n";
    }
}
```



```
        echo "End of page one of results. Alter the \$pageLength variable to see more
        results.\n";
        break;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListKeys](#)」を参照してください。

コードの例

- [AWS SDKs AWS KMS を使用するための基本的な例](#)
 - [こんにちは AWS Key Management Serviceは](#)
 - [AWS SDK AWS KMS を使用した の基本について説明します。](#)
 - [AWS SDKs AWS KMS を使用するためのアクション](#)
 - [AWS SDK または CLI CreateAliasで を使用する](#)
 - [AWS SDK または CLI CreateGrantで を使用する](#)
 - [AWS SDK または CLI CreateKeyで を使用する](#)
 - [AWS SDK または CLI Decryptで を使用する](#)
 - [AWS SDK または CLI DeleteAliasで を使用する](#)
 - [AWS SDK または CLI DescribeKeyで を使用する](#)
 - [AWS SDK または CLI DisableKeyで を使用する](#)
 - [AWS SDK または CLI EnableKeyで を使用する](#)
 - [AWS SDK または CLI EnableKeyRotationで を使用する](#)
 - [AWS SDK または CLI Encryptで を使用する](#)
 - [AWS SDK または CLI GenerateDataKeyで を使用する](#)
 - [AWS SDK または CLI GenerateDataKeyWithoutPlaintextで を使用する](#)
 - [AWS SDK または CLI GenerateRandomで を使用する](#)
 - [AWS SDK または CLI GetKeyPolicyで を使用する](#)
 - [AWS SDK または CLI ListAliasesで を使用する](#)
 - [AWS SDK または CLI ListGrantsで を使用する](#)
 - [AWS SDK または CLI ListKeyPoliciesで を使用する](#)
 - [AWS SDK または CLI ListKeysで を使用する](#)

- [AWS SDK または CLI PutKeyPolicyで を使用する](#)
- [AWS SDK または CLI ReEncryptで を使用する](#)
- [AWS SDK または CLI RetireGrantで を使用する](#)
- [AWS SDK または CLI RevokeGrantで を使用する](#)
- [AWS SDK または CLI ScheduleKeyDeletionで を使用する](#)
- [AWS SDK または CLI Signで を使用する](#)
- [AWS SDK または CLI TagResourceで を使用する](#)
- [AWS SDK または CLI UpdateAliasで を使用する](#)
- [AWS SDK または CLI Verifyで を使用する](#)
- [AWS SDKs AWS KMS を使用するシナリオ](#)
 - [v2 を使用した DynamoDB AWS Command Line Interface テーブル暗号化の操作](#)

AWS SDKs AWS KMS を使用するための基本的な例

次のコード例は、SDKs AWS Key Management Service で AWS の基本を使用する方法を示しています。

例

- [こんにちは AWS Key Management Serviceは](#)
- [AWS SDK AWS KMS を使用した の基本について説明します。](#)
- [AWS SDKs AWS KMS を使用するためのアクション](#)
 - [AWS SDK または CLI CreateAliasで を使用する](#)
 - [AWS SDK または CLI CreateGrantで を使用する](#)
 - [AWS SDK または CLI CreateKeyで を使用する](#)
 - [AWS SDK または CLI Decryptで を使用する](#)
 - [AWS SDK または CLI DeleteAliasで を使用する](#)
 - [AWS SDK または CLI DescribeKeyで を使用する](#)
 - [AWS SDK または CLI DisableKeyで を使用する](#)
 - [AWS SDK または CLI EnableKeyで を使用する](#)
 - [AWS SDK または CLI EnableKeyRotationで を使用する](#)

- [AWS SDK または CLI GenerateDataKeyで を使用する](#)
- [AWS SDK または CLI GenerateDataKeyWithoutPlaintextで を使用する](#)
- [AWS SDK または CLI GenerateRandomで を使用する](#)
- [AWS SDK または CLI GetKeyPolicyで を使用する](#)
- [AWS SDK または CLI ListAliasesで を使用する](#)
- [AWS SDK または CLI ListGrantsで を使用する](#)
- [AWS SDK または CLI ListKeyPoliciesで を使用する](#)
- [AWS SDK または CLI ListKeysで を使用する](#)
- [AWS SDK または CLI PutKeyPolicyで を使用する](#)
- [AWS SDK または CLI ReEncryptで を使用する](#)
- [AWS SDK または CLI RetireGrantで を使用する](#)
- [AWS SDK または CLI RevokeGrantで を使用する](#)
- [AWS SDK または CLI ScheduleKeyDeletionで を使用する](#)
- [AWS SDK または CLI Signで を使用する](#)
- [AWS SDK または CLI TagResourceで を使用する](#)
- [AWS SDK または CLI UpdateAliasで を使用する](#)
- [AWS SDK または CLI Verifyで を使用する](#)

こんにちは AWS Key Management Serviceは

次のコード例は、AWS Key Management Serviceの使用を開始する方法を示しています。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;  
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
```

```
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * 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
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /**
         * The `subscribe` method is required when using paginator methods in the
         * AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
         * which is
         * based on a reactive stream. This allows asynchronous retrieval of
         * paginated
         * results as they become available. By subscribing to the stream, we can
         * process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
            kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key
                Id is: " + key.keyId()))
            .whenComplete((result, exception) -> {
                if (exception != null) {
```

```
        System.err.println("Error occurred: " +
exception.getMessage());
    } else {
        System.out.println("Successfully listed all keys.");
    }
});

try {
    future.join();
} catch (Exception e) {
    System.err.println("Failed to list keys: " + e.getMessage());
}
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListKeys](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
include "vendor/autoload.php";

use Aws\Kms\KmsClient;

echo "This file shows how to connect to the KmsClient, uses a paginator to get
the keys for the account, and lists the KeyIds for up to 10 keys.\n";

$client = new KmsClient([]);

$pageLength = 10; // Change this value to change the number of records shown, or
to break up the result into pages.
```

```
$keys = [];  
$keysPaginator = $client->getPaginator("ListKeys", ['Limit' => $pageLength]);  
foreach($keysPaginator as $page){  
    foreach($page['Keys'] as $index => $key){  
        echo "The $index index Key's ID is: {$key['KeyId']}\n";  
    }  
    echo "End of page one of results. Alter the \$pageLength variable to see more  
results.\n";  
    break;  
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListKeys](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK AWS KMS を使用した の基本について説明します。

次のコード例は、以下を実行する方法を示しています。

- KMS キーを作成します。
- アカウントの KMS キーを一覧表示し、詳細を取得します。
- KMS キーを有効および無効にします。
- クライアント側の暗号化に使用できる対称データキーを生成します。
- データのデジタル署名に使用する非対称キーを生成します。
- キーをタグ付けします。
- KMS キーを削除します。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトでシナリオを実行します。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * 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
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
        "-");
    private static String accountId = "";
```

```
private static final Logger logger =
LoggerFactory.getLogger(KMSScenario.class);

static KMSActions kmsActions = new KMSActions();

static Scanner scanner = new Scanner(System.in);

static String aliasName = "alias/dev-encryption-key";

public static void main(String[] args) {
    final String usage = ""
        Usage: <granteePrincipal>

        Where:
            granteePrincipal - The principal (user, service account, or group)
to whom the grant or permission is being given.
        """;

    if (args.length != 1) {
        logger.info(usage);
        return;
    }
    String granteePrincipal = args[0];
    String policyName = "default";

    accountId = kmsActions.getAccountId();
    String keyDesc = "Created by the AWS KMS API";

    logger.info(DASHES);
    logger.info("")
        Welcome to the AWS Key Management SDK Basics scenario.

        This program demonstrates how to interact with AWS Key Management
using the AWS SDK for Java (v2).
        The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
            and manage AWS KMS keys and control their use across a wide range of
AWS services and applications.
        KMS provides a centralized and unified approach to managing
encryption keys, making it easier to meet your
            data protection and regulatory compliance requirements.

        This Basics scenario creates two key types:
```


- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...

```
""");
waitForInputToContinue(scanner);

try {
    // Run the methods that belong to this scenario.
    String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
    requestDeleteResources(aliasName, targetKeyId);

} catch (Throwable rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof KmsException kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
}

private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
    logger.info(DASHES);
    logger.info("1. Create a symmetric KMS key\n");
    logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
    waitForInputToContinue(scanner);
    String targetKeyId;
    try {
        CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
        targetKeyId = futureKeyId.join();
        logger.info("A symmetric key was successfully created " +
targetKeyId);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {

```

```

        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
    throw cause;
}
waitForInputToContinue(scanner);

logger.info(DASHES);
logger.info("
    2. Enable a KMS key

    By default, when the SDK creates an AWS key, it is enabled. The next
    bit of code checks to
    determine if the key is enabled.
    ");
waitForInputToContinue(scanner);
boolean isEnabled;
try {
    CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
    isEnabled = futureIsKeyEnabled.join();
    logger.info("Is the key enabled? {}", isEnabled);

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof KmsException kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
    throw cause;
}

if (!isEnabled)
    try {
        CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());

```

```

        } else {
            logger.info("An unexpected error occurred: " +
rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);

    logger.info(DASHES);
    logger.info("3. Encrypt data using the symmetric KMS key");
    String plaintext = "Hello, AWS KMS!";
    logger.info(""""
        One of the main uses of symmetric keys is to encrypt and decrypt
data.
        Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
        """, plaintext);
    waitForInputToContinue(scanner);
    SdkBytes encryptedData;
    try {
        CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
        encryptedData = future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof DisabledException kmsDisabledEx) {
            logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        deleteKey(targetKeyId);
        throw cause;
    }
    waitForInputToContinue(scanner);

    logger.info(DASHES);
    logger.info("4. Create an alias");
    logger.info(""""
        The alias name should be prefixed with 'alias/'.
        The default, 'alias/dev-encryption-key'.

```

```
        "");
    waitForInputToContinue(scanner);

    try {
        CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof AlreadyExistsException kmsExistsEx) {
            if (kmsExistsEx.getMessage().contains("already exists")) {
                logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
            }
        } else {
            logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);

            deleteKey(targetKeyId);
            throw cause;
        }
    }
    waitForInputToContinue(scanner);

    logger.info(DASHES);
    logger.info("5. List all of your aliases");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        deleteAliasName(aliasName);
        deleteKey(targetKeyId);
        throw cause;
    }
    waitForInputToContinue(scanner);
```

```

logger.info(DASHES);
logger.info("6. Enable automatic rotation of the KMS key");
logger.info("""

    By default, when the SDK enables automatic rotation of a KMS key,
    KMS rotates the key material of the KMS key one year (approximately
    365 days) from the enable date and every year
    thereafter.
    """);
waitForInputToContinue(scanner);
try {
    CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
    future.join();

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof KmsException kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
    deleteAliasName(aliasName);
    deleteKey(targetKeyId);
    throw cause;
}
waitForInputToContinue(scanner);

logger.info(DASHES);
logger.info("""
    7. Create a grant

    A grant is a policy instrument that allows Amazon Web Services
    principals to use KMS keys.
    It also can allow them to view a KMS key (DescribeKey) and create and
    manage grants.
    When authorizing access to a KMS key, grants are considered along
    with key policies and IAM policies.
    """);

waitForInputToContinue(scanner);
String grantId = null;

```

```
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info(""
```

The revocation of a grant immediately removes the permissions and access that the grant had provided.

This means that any principal (user, role, or service) that was granted access to perform specific

KMS operations on a KMS key will no longer be able to perform those operations.

```

        """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
            earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {

```

```

        CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
        decryptedData = future.join();
        logger.info("Decrypted data: " + decryptedData);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        deleteAliasName(aliasName);
        deleteKey(targetKeyId);
        throw cause;
    }
    logger.info("Decrypted text is: " + decryptedData);
    waitForInputToContinue(scanner);

    logger.info(DASHES);
    logger.info("11. Replace a key policy\n");
    logger.info(""""
        A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
        access to KMS keys. Every KMS key must have exactly one key policy.
The statements in the key policy
        determine who has permission to use the KMS key and how they can use
it.

        You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
        must have a key policy.

        By default, when you create a key by using the SDK, a policy is
created that
        gives the AWS account that owns the KMS key full access to the KMS
key.

        Let's try to replace the automatically created policy with the
following policy.

        "Version": "2012-10-17",
        "Statement": [{
        "Effect": "Allow",

```



```

        "Principal": {"AWS": "arn:aws:iam::000000000000:root"},
        "Action": "kms:*",
        "Resource": "*"
    }]
    """);

    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
        boolean success = future.join();
        if (success) {
            logger.info("Key policy replacement succeeded.");
        } else {
            logger.error("Key policy replacement failed.");
        }
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        deleteAliasName(aliasName);
        deleteKey(targetKeyId);
        throw cause;
    }
    waitForInputToContinue(scanner);

    logger.info(DASHES);
    logger.info("12. Get the key policy\n");
    logger.info("The next bit of code that runs gets the key policy to make
sure it exists.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
        String policy = future.join();
        if (!policy.isEmpty()) {
            logger.info("Retrieved policy: " + policy);
        }
    }

```

```

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        deleteAliasName(aliasName);
        deleteKey(targetKeyId);
        throw cause;
    }
    waitForInputToContinue(scanner);

    logger.info(DASHES);
    logger.info("13. Create an asymmetric KMS key and sign your data\n");
    logger.info("""
        Signing your data with an AWS key can provide several benefits that
make it an attractive option
        for your data signing needs. By using an AWS KMS key, you can
leverage the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance
the overall security posture
        of your organization.
        """);
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
        boolean success = future.join();
        if (success) {
            logger.info("Sign and verify data operation succeeded.");
        } else {
            logger.error("Sign and verify data operation failed.");
        }
    }

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
    }

```

```

        deleteAliasName(aliasName);
        deleteKey(targetKeyId);
        throw cause;
    }
    waitForInputToContinue(scanner);

    logger.info(DASHES);
    logger.info("14. Tag your symmetric KMS Key\n");
    logger.info("
        By using tags, you can improve the overall management, security, and
        governance of your
        KMS keys, making it easier to organize, track, and control access to
        your encrypted data within
        your AWS environment
        ");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
kmsActions.tagKMSKeyAsync(targetKeyId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        deleteAliasName(aliasName);
        deleteKey(targetKeyId);
        throw cause;
    }
    waitForInputToContinue(scanner);
    return targetKeyId;
}

// Deletes KMS resources with user input.
private static void requestDeleteResources(String aliasName, String
targetKeyId) {
    logger.info(DASHES);
    logger.info("15. Schedule the deletion of the KMS key\n");
    logger.info("
        By default, KMS applies a waiting period of 30 days,

```

but you can specify a waiting period of 7-30 days. When this operation is successful, the key state of the KMS key changes to PendingDeletion and the key can't be used in any cryptographic operations. It remains in this state for the duration of the waiting period.

Deleting a KMS key is a destructive and potentially dangerous operation. When a KMS key is deleted, all data that was encrypted under the KMS key is unrecoverable.

```
""");
logger.info("Would you like to delete the Key Management resources? (y/n)");

String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    logger.info("You selected to delete the AWS KMS resources.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " +
rt.getMessage());
        }
    }
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
```

```
        logger.info("An unexpected error occurred: " +
rt.getMessage());
    }
}

    try {
        CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error
code {}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " +
rt.getMessage());
        }
    }

    } else {
        logger.info("The Key Management resources will not be deleted");
    }

    logger.info(DASHES);
    logger.info("This concludes the AWS Key Management SDK scenario");
    logger.info(DASHES);
}

// This method is invoked from Exceptions to clean up the resources.
private static void deleteKey(String targetKeyId) {
    try {
        CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
    }
}
```

```
        }
    }

    try {
        CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
    }
}

// This method is invoked from Exceptions to clean up the resources.
private static void deleteAliasName(String aliasName) {
    try {
        CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof KmsException kmsEx) {
            logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
    }
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
```

```

        logger.info("Continuing with the program...");
        logger.info("");
        break;
    } else {
        // Handle invalid input.
        logger.info("Invalid input. Please try again.");
    }
}
}
}
}

```

KMS アクションをラップするクラスを定義します。

```

public class KMSActions {
    private static final Logger logger =
        LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async
     * client, with the following configurations:
     * <ul>
     * <li>Max concurrency: 100</li>
     * <li>Connection timeout: 60 seconds</li>
     * <li>Read timeout: 60 seconds</li>
     * <li>Write timeout: 60 seconds</li>
     * <li>API call timeout: 2 minutes</li>
     * <li>API call attempt timeout: 90 seconds</li>
     * <li>Retry policy: up to 3 retries</li>
     * <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
     * creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()

```

```

        .maxConcurrency(100)
        .connectionTimeout(Duration.ofSeconds(60))
        .readTimeout(Duration.ofSeconds(60))
        .writeTimeout(Duration.ofSeconds(60))
        .build();

    ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))
        .apiCallAttemptTimeout(Duration.ofSeconds(90))
        .retryPolicy(RetryPolicy.builder()
            .numRetries(3)
            .build())
        .build();

    kmsAsyncClient = KmsAsyncClient.builder()
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
}
return kmsAsyncClient;
}

/**
 * Creates a new symmetric encryption key asynchronously.
 *
 * @param keyDesc the description of the key to be created
 * @return a {@link CompletableFuture} that completes with the ID of the
newly created key
 * @throws RuntimeException if an error occurs while creating the key
 */
public CompletableFuture<String> createKeyAsync(String keyDesc) {
    CreateKeyRequest keyRequest = CreateKeyRequest.builder()
        .description(keyDesc)
        .keySpec(KeySpec.SYMMETRIC_DEFAULT)
        .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
        .build();

    return getAsyncClient().createKey(keyRequest)
        .thenApply(resp -> resp.keyMetadata().keyId())
        .exceptionally(ex -> {
            throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
        });
}

```



```

    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates
     * whether the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key
     * state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
            getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}",
keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
     * enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)

```

```

        .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and
key ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
            }
        });
    }

```

```

        logger.info("The string was encrypted with algorithm {}.\"",
algorithm);
    } else {
        throw new RuntimeException(ex);
    }
}).thenApply(EncryptResponse::ciphertextBlob);
}

/**
 * Creates a custom alias for the specified target key asynchronously.
 *
 * @param targetKeyId the ID of the target key for the alias
 * @param aliasName the name of the alias to create
 * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
 */
public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
    CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
        .aliasName(aliasName)
        .targetKeyId(targetKeyId)
        .build();

    CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
    responseFuture.whenComplete((response, exception) -> {
        if (exception == null) {
            logger.info("{} was successfully created.", aliasName);
        } else {
            if (exception instanceof ResourceExistsException) {
                logger.info("Alias [{}] already exists. Moving on...",
aliasName);
            } else if (exception instanceof KmsException kmsEx) {
                throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
            } else {
                throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
            }
        }
    });

    return responseFuture.thenApply(response -> null);
}

```

```

/**
 * Asynchronously lists all the aliases in the current AWS account.
 *
 * @return a {@link CompletableFuture} that completes when the list of
aliases has been processed
 */
public CompletableFuture<Object> listAllAliasesAsync() {
    ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
        .limit(15)
        .build();

    ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
    return paginator.subscribe(response -> {
        response.aliases().forEach(alias ->
            logger.info("The alias name is: " + alias.aliasName())
        );
    })
        .thenApply(v -> null)
        .exceptionally(ex -> {
            if (ex.getCause() instanceof KmsException) {
                KmsException e = (KmsException) ex.getCause();
                throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
            } else {
                throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
            }
        });
}

/**
 * Enables key rotation asynchronously for the specified key ID.
 *
 * @param keyId the ID of the key for which to enable key rotation
 * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
 * @throws RuntimeException if there was an error enabling key rotation,
either due to a KMS exception or an unexpected error
 */
public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {

```

```

        EnableKeyRotationRequest enableKeyRotationRequest =
        EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyRotationResponse> responseFuture =
        getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key rotation has been enabled for key with id [{}]",
                    keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to enable key rotation: "
+ kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
                        exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }

    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
     asynchronously.
     *
     * @param keyId          The unique identifier for the customer master key
     (CMK) that the grant applies to.
     * @param granteePrincipal The principal that is given permission to perform
     the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID
     of the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
     process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
        granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY

```

```

    );

    CreateGrantRequest grantRequest = CreateGrantRequest.builder()
        .keyId(keyId)
        .name("grant1")
        .granteePrincipal(granteePrincipal)
        .operations(grantPermissions)
        .build();

    CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
    responseFuture.whenComplete((response, ex) -> {
        if (ex == null) {
            logger.info("Grant created successfully with ID: " +
response.grantId());
        } else {
            if (ex instanceof KmsException kmsEx) {
                throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
            } else {
                throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }
    });

    return responseFuture.thenApply(CreateGrantResponse::grantId);
}

/**
 * Asynchronously displays the grant IDs for the specified key ID.
 *
 * @param keyId the ID of the AWS KMS key for which to list the grants
 * @return a {@link CompletableFuture} that, when completed, will be null
if the operation succeeded, or will throw a {@link RuntimeException} if the
operation failed
 * @throws RuntimeException if there was an error listing the grants, either
due to an {@link KmsException} or an unexpected error
 */
public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
    ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
        .keyId(keyId)
        .limit(15)
        .build();

```

```

        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
            response.grants().forEach(grant -> {
                logger.info("The grant Id is: " + grant.grantId());
            });
        })
        .thenApply(v -> null)
        .exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof KmsException) {
                throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
            } else {
                throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
            }
        });
    }

    /**
     * Revokes a grant for the specified AWS KMS key asynchronously.
     *
     * @param keyId    The ID or key ARN of the AWS KMS key.
     * @param grantId  The identifier of the grant to be revoked.
     * @return A {@link CompletableFuture} representing the asynchronous
operation of revoking the grant.
     *
     * The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
     *
     * if the operation is successful, or with a {@code null} value if an
error occurs.
     */
    public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {

```

```

        logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
    } else {
        if (exception instanceof KmsException kmsEx) {
            if (kmsEx.getMessage().contains("Grant does not exist")) {
                logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
            } else {
                throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
            }
        } else {
            throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
        }
    }
});

return responseFuture;
}

/**
 * Asynchronously decrypts the given encrypted data using the specified key
ID.
 *
 * @param encryptedData The encrypted data to be decrypted.
 * @param keyId The ID of the key to be used for decryption.
 * @return A CompletableFuture that, when completed, will contain the
decrypted data as a String.
 *
 * If an error occurs during the decryption process, the
CompletableFuture will complete
 *
 * exceptionally with the error, and the method will return an empty
String.
 */
public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
    DecryptRequest decryptRequest = DecryptRequest.builder()
        .ciphertextBlob(encryptedData)
        .keyId(keyId)
        .build();

    CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);

```



```

        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId      the ID of the KMS key to replace the policy for
     * @param policyName the name of the policy to be replaced
     * @param accountId  the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean
indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = ""
        {
            "Version": "2012-10-17",
            "Statement": [{
                "Effect": "Allow",
                "Principal": {"AWS": "arn:aws:iam::%s:root"},
                "Action": "kms:*",
                "Resource": "*"
            }]
        }
        """".formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)

```

```

        .policyName(policyName)
        .policy(policy)
        .build();

// First, get the current policy to check if it exists
return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
    .thenCompose(response -> {
        logger.info("Current policy exists. Replacing it...");
        return getAsyncClient().putKeyPolicy(keyPolicyRequest);
    })
    .thenApply(putPolicyResponse -> {
        logger.info("The key policy has been replaced.");
        return true;
    })
    .exceptionally(throwable -> {
        if (throwable.getCause() instanceof LimitExceededException) {
            logger.error("Cannot replace policy, as only one policy is
allowed per key.");
            return false;
        }
        throw new RuntimeException("Error replacing policy", throwable);
    });
}

/**
 * Asynchronously retrieves the key policy for the specified key ID and
 * policy name.
 *
 * @param keyId      the ID of the AWS KMS key for which to retrieve the
 * policy
 * @param policyName the name of the key policy to retrieve
 * @return a {@link CompletableFuture} that, when completed, contains the key
 * policy as a {@link String}
 */
public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
    GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
        .keyId(keyId)
        .policyName(policyName)
        .build();

    return getAsyncClient().getKeyPolicy(policyRequest)

```

```

        .thenApply(response -> {
            String policy = response.policy();
            logger.info("The response is: " + policy);
            return policy;
        })
        .exceptionally(ex -> {
            throw new RuntimeException("Failed to get key policy", ex);
        });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
     *     origin.</li>
     *     <li>Signs the provided message using the created KMS key and the
     *     RSASSA-PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key
     *     and the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
     * signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();
            });
    }

```

```

        SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
        SignRequest signRequest = SignRequest.builder()
            .keyId(keyId)
            .message(messageBytes)
            .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
            .build();

        return getAsyncClient().sign(signRequest)
            .thenCompose(signResponse -> {
                byte[] signedBytes =
signResponse.signature().asByteArray();

                VerifyRequest verifyRequest = VerifyRequest.builder()
                    .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().verify(verifyRequest)
                    .thenApply(verifyResponse -> {
                        return (boolean) verifyResponse.signatureValid();
                    });
            });
    })
    .exceptionally(throwable -> {
        throw new RuntimeException("Failed to sign or verify data",
throwable);
    });
}

/**
 * Asynchronously tags a KMS key with a specific tag.
 *
 * @param keyId the ID of the KMS key to be tagged
 * @return a {@link CompletableFuture} that completes when the tagging
operation is finished
 */
public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
    Tag tag = Tag.builder()

```

```

        .tagKey("Environment")
        .tagValue("Production")
        .build();

    TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
        .keyId(keyId)
        .tags(tag)
        .build();

    return getAsyncClient().tagResource(tagResourceRequest)
        .thenRun(() -> {
            logger.info("{} key was tagged", keyId);
        })
        .exceptionally(throwable -> {
            throw new RuntimeException("Failed to tag the KMS key",
throwable);
        });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous
operation of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
            });
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS)
key.

```

```

    *
    * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
    disabled
    * @return a CompletableFuture that, when completed, indicates that the key
    has been disabled successfully
    */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
                    throwable);
            });
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
     potentially dangerous operation.
     * When a KMS key is deleted, all data that was encrypted under the KMS key
     becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted
     using the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when
     deleting KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion
     is scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
        ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();
    }

```

```
        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for
key ID: " + keyId, throwable);
            });
    }

    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- APIの詳細については、『AWS SDK for Java 2.x API リファレンス』の以下のトピックを参照してください。

- [CreateAlias](#)
- [CreateGrant](#)
- [CreateKey](#)
- [Decrypt](#)
- [DescribeKey](#)
- [DisableKey](#)
- [EnableKey](#)
- [暗号化](#)
- [GetKeyPolicy](#)
- [ListAliases](#)
- [ListGrants](#)
- [ListKeys](#)
- [RevokeGrant](#)
- [ScheduleKeyDeletion](#)

- [Sign](#)
- [TagResource](#)

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
echo "\n";
echo "-----\n";
echo <<<WELCOME
Welcome to the AWS Key Management Service SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management Service using
the AWS SDK for PHP (v3).
The AWS Key Management Service (KMS) is a secure and highly available service
that allows you to create
and manage AWS KMS keys and control their use across a wide range of AWS services
and applications.
KMS provides a centralized and unified approach to managing encryption keys,
making it easier to meet your
data protection and regulatory compliance requirements.

This KMS Basics scenario creates two key types:
- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...\n
WELCOME;

echo "-----\n";
$this->pressEnter();

$this->kmsClient = new KmsClient([]);
// Initialize the KmsService class with the client. This allows you to
override any defaults in the client before giving it to the service class.
```



```
$this->kmsService = new KmsService($this->kmsClient);

// 1. Create a symmetric KMS key.
echo "\n";
echo "1. Create a symmetric KMS key.\n";
echo "First, we will create a symmetric KMS key that is used to encrypt
and decrypt data by invoking createKey().\n";
$this->pressEnter();

$key = $this->kmsService->createKey();
$this->resources['symmetricKey'] = $key['KeyId'];
echo "Created a customer key with ARN {$key['Arn']}. \n";
$this->pressEnter();

// 2. Enable a KMS key.
echo "\n";
echo "2. Enable a KMS key.\n";
echo "By default when you create an AWS key, it is enabled. The code
checks to
determine if the key is enabled. If it is not enabled, the code enables it.\n";
$this->pressEnter();

$keyInfo = $this->kmsService->describeKey($key['KeyId']);
if(!$keyInfo['Enabled']){
    echo "The key was not enabled, so we will enable it.\n";
    $this->pressEnter();
    $this->kmsService->enableKey($key['KeyId']);
    echo "The key was successfully enabled.\n";
}else{
    echo "The key was already enabled, so there was no need to enable it.
\n";
}
$this->pressEnter();

// 3. Encrypt data using the symmetric KMS key.
echo "\n";
echo "3. Encrypt data using the symmetric KMS key.\n";
echo "One of the main uses of symmetric keys is to encrypt and decrypt
data.\n";
echo "Next, we'll encrypt the string 'Hello, AWS KMS!' with the
SYMMETRIC_DEFAULT encryption algorithm.\n";
$this->pressEnter();
$text = "Hello, AWS KMS!";
$encryption = $this->kmsService->encrypt($key['KeyId'], $text);
```

```
    echo "The plaintext data was successfully encrypted with the algorithm:
{$encryption['EncryptionAlgorithm']}\n";
    $this->pressEnter();

    // 4. Create an alias.
    echo "\n";
    echo "4. Create an alias.\n";
    $aliasInput = testable_readline("Please enter an alias prefixed with
\"alias/\" or press enter to use a default value: ");
    if($aliasInput == ""){
        $aliasInput = "alias/dev-encryption-key";
    }
    $this->kmsService->createAlias($key['KeyId'], $aliasInput);
    $this->resources['alias'] = $aliasInput;
    echo "The alias \"$aliasInput\" was successfully created.\n";
    $this->pressEnter();

    // 5. List all of your aliases.
    $aliasPageSize = 10;
    echo "\n";
    echo "5. List all of your aliases, up to $aliasPageSize.\n";
    $this->pressEnter();
    $aliasPaginator = $this->kmsService->listAliases();
    foreach($aliasPaginator as $pages){
        foreach($pages['Aliases'] as $alias){
            echo $alias['AliasName'] . "\n";
        }
        break;
    }
    $this->pressEnter();

    // 6. Enable automatic rotation of the KMS key.
    echo "\n";
    echo "6. Enable automatic rotation of the KMS key.\n";
    echo "By default, when the SDK enables automatic rotation of a KMS key,
KMS rotates the key material of the KMS key one year (approximately 365 days)
from the enable date and every year
thereafter.";
    $this->pressEnter();
    $this->kmsService->enableKeyRotation($key['KeyId']);
    echo "The key's rotation was successfully set for key:
{$key['KeyId']}\n";
    $this->pressEnter();
```

```
// 7. Create a grant.
echo "7. Create a grant.\n";
echo "\n";
echo "A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
It also can allow them to view a KMS key (DescribeKey) and create and manage
grants.
When authorizing access to a KMS key, grants are considered along with key
policies and IAM policies.\n";
$granteeARN = testable_readline("Please enter the Amazon Resource Name
(ARN) of an Amazon Web Services principal. Valid principals include Amazon
Web Services accounts, IAM users, IAM roles, federated users, and assumed
role users. For help with the ARN syntax for a principal, see IAM ARNs in the
Identity and Access Management User Guide. \nTo skip this step, press enter
without any other values: ");
if($granteeARN){
    $operations = [
        "ENCRYPT",
        "DECRYPT",
        "DESCRIBE_KEY",
    ];
    $grant = $this->kmsService->createGrant($key['KeyId'], $granteeARN,
$operations);
    echo "The grant Id is: {$grant['GrantId']}\n";
}else{
    echo "Steps 7, 8, and 9 will be skipped.\n";
}
$this->pressEnter();

// 8. List grants for the KMS key.
if($granteeARN){
    echo "8. List grants for the KMS key.\n\n";
    $grantsPaginator = $this->kmsService->listGrants($key['KeyId']);
    foreach($grantsPaginator as $page){
        foreach($page['Grants'] as $grant){
            echo $grant['GrantId'] . "\n";
        }
    }
}else{
    echo "Skipping step 8...\n";
}
$this->pressEnter();

// 9. Revoke the grant.
```

```

        if($granteeARN) {
            echo "\n";
            echo "9. Revoke the grant.\n";
            $this->pressEnter();
            $this->kmsService->revokeGrant($grant['GrantId'], $keyInfo['KeyId']);
            echo "{$grant['GrantId']} was successfully revoked!\n";
        }else{
            echo "Skipping step 9...\n";
        }
        $this->pressEnter();

        // 10. Decrypt the data.
        echo "\n";
        echo "10. Decrypt the data.\n";
        echo "Let's decrypt the data that was encrypted before.\n";
        echo "We'll use the same key to decrypt the string that we encrypted
earlier in the program.\n";
        $this->pressEnter();
        $decryption = $this->kmsService->decrypt($keyInfo['KeyId'],
$encryption['CiphertextBlob'], $encryption['EncryptionAlgorithm']);
        echo "The decrypted text is: {$decryption['Plaintext']}\n";
        $this->pressEnter();

        // 11. Replace a Key Policy.
        echo "\n";
        echo "11. Replace a Key Policy.\n";
        echo "A key policy is a resource policy for a KMS key. Key policies are
the primary way to control access to KMS keys.\n";
        echo "Every KMS key must have exactly one key policy. The statements in
the key policy determine who has permission to use the KMS key and how they can
use it.\n";
        echo " You can also use IAM policies and grants to control access to the
KMS key, but every KMS key must have a key policy.\n";
        echo "We will replace the key's policy with a new one:\n";
        $stsClient = new StsClient([]);
        $result = $stsClient->getCallerIdentity();
        $accountId = $result['Account'];
        $keyPolicy = <<< KEYPOLICY
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::$accountId:root"},
        "Action": "kms:*",

```

```

        "Resource": "*"
    }]
}
KEYPOLICY;
    echo $keyPolicy;
    $this->pressEnter();
    $this->kmsService->putKeyPolicy($keyInfo['KeyId'], $keyPolicy);
    echo "The Key Policy was successfully replaced!\n";
    $this->pressEnter();

    // 12. Retrieve the key policy.
    echo "\n";
    echo "12. Retrieve the key policy.\n";
    echo "Let's get some information about the new policy and print it to the
screen.\n";
    $this->pressEnter();
    $policyInfo = $this->kmsService->getKeyPolicy($keyInfo['KeyId']);
    echo "We got the info! Here is the policy: \n";
    echo $policyInfo['Policy'] . "\n";
    $this->pressEnter();

    // 13. Create an asymmetric KMS key and sign data.
    echo "\n";
    echo "13. Create an asymmetric KMS key and sign data.\n";
    echo "Signing your data with an AWS key can provide several benefits that
make it an attractive option for your data signing needs.\n";
    echo "By using an AWS KMS key, you can leverage the security controls and
compliance features provided by AWS, which can help you meet various regulatory
requirements and enhance the overall security posture of your organization.\n";
    echo "First we'll create the asymmetric key.\n";
    $this->pressEnter();
    $keySpec = "RSA_2048";
    $keyUsage = "SIGN_VERIFY";
    $asymmetricKey = $this->kmsService->createKey($keySpec, $keyUsage);
    $this->resources['asymmetricKey'] = $asymmetricKey['KeyId'];
    echo "Created the key with ID: {$asymmetricKey['KeyId']}\n";
    echo "Next, we'll sign the data.\n";
    $this->pressEnter();
    $algorithm = "RSASSA_PSS_SHA_256";
    $sign = $this->kmsService->sign($asymmetricKey['KeyId'], $text,
$algorithm);
    $verify = $this->kmsService->verify($asymmetricKey['KeyId'], $text,
$sign['Signature'], $algorithm);
    echo "Signature verification result: {$sign['signature']}\n";

```

```

$this->pressEnter();

// 14. Tag the symmetric KMS key.
echo "\n";
echo "14. Tag the symmetric KMS key.\n";
echo "By using tags, you can improve the overall management, security,
and governance of your KMS keys, making it easier to organize, track, and
control access to your encrypted data within your AWS environment.\n";
echo "Let's tag our symmetric key as Environment->Production\n";
$this->pressEnter();
$this->kmsService->tagResource($key['KeyId'], [
    [
        'TagKey' => "Environment",
        'TagValue' => "Production",
    ],
]);
echo "The key was successfully tagged!\n";
$this->pressEnter();

// 15. Schedule the deletion of the KMS key
echo "\n";
echo "15. Schedule the deletion of the KMS key.\n";
echo "By default, KMS applies a waiting period of 30 days, but you can
specify a waiting period of 7-30 days.\n";
echo "When this operation is successful, the key state of the KMS key
changes to PendingDeletion and the key can't be used in any cryptographic
operations.\n";
echo "It remains in this state for the duration of the waiting period.\n
\n";

echo "Deleting a KMS key is a destructive and potentially dangerous
operation. When a KMS key is deleted, all data that was encrypted under the KMS
key is unrecoverable.\n\n";

$cleanUp = testable_readline("Would you like to delete the resources
created during this scenario, including the keys? (y/n): ");
if($cleanUp == "Y" || $cleanUp == "y"){
    $this->cleanUp();
}

echo
"-----
\n";

echo "This concludes the AWS Key Management SDK Basics scenario\n";

```

```
        echo
        "-----\n";

namespace Kms;

use Aws\Kms\Exception\KmsException;
use Aws\Kms\KmsClient;
use Aws\Result;
use Aws\ResultPaginator;
use AwsUtilities\AWSServiceClass;

class KmsService extends AWSServiceClass
{
    protected KmsClient $client;
    protected bool $verbose;

    /**
     * @param KmsClient|null $client
     * @param bool $verbose
     */
    public function __construct(KmsClient $client = null, bool $verbose = false)
    {
        $this->verbose = $verbose;
        if($client){
            $this->client = $client;
            return;
        }
        $this->client = new KmsClient([]);
    }

    /**
     * @param string $keySpec
     * @param string $keyUsage
     * @param string $description
     * @return array
     */
    public function createKey(string $keySpec = "", string $keyUsage = "", string
    $description = "Created by the SDK for PHP")
    {

```

```

        $parameters = ['Description' => $description];
        if($keySpec && $keyUsage){
            $parameters['KeySpec'] = $keySpec;
            $parameters['KeyUsage'] = $keyUsage;
        }
        try {
            $result = $this->client->createKey($parameters);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            // Check for error specific to createKey operations
            if ($caught->getAwsErrorMessage() == "LimitExceededException"){
                echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
            }
            throw $caught;
        }
    }

    /**
     * @param string $keyId
     * @param string $ciphertext
     * @param string $algorithm
     * @return Result
     */
    public function decrypt(string $keyId, string $ciphertext, string $algorithm
= "SYMMETRIC_DEFAULT")
    {
        try{
            return $this->client->decrypt([
                'CiphertextBlob' => $ciphertext,
                'EncryptionAlgorithm' => $algorithm,
                'KeyId' => $keyId,
            ]);
        }catch(KmsException $caught){
            echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
}

```



```
/**
 * @param string $keyId
 * @param string $text
 * @return Result
 */
public function encrypt(string $keyId, string $text)
{
    try {
        return $this->client->encrypt([
            'KeyId' => $keyId,
            'Plaintext' => $text,
        ]);
    } catch (KmsException $caught){
        if($caught->getAwsErrorMessage() == "DisabledException"){
            echo "The request was rejected because the specified KMS key is
not enabled.\n";
        }
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param int $limit
 * @return ResultPaginator
 */
public function listAliases(string $keyId = "", int $limit = 0)
{
    $args = [];
    if($keyId){
        $args['KeyId'] = $keyId;
    }
    if($limit){
        $args['Limit'] = $limit;
    }
    try{
        return $this->client->getPaginator("ListAliases", $args);
    } catch (KmsException $caught){
        if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
            echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
        }
    }
}
```

```
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param string $alias
 * @return void
 */
public function createAlias(string $keyId, string $alias)
{
    try{
        $this->client->createAlias([
            'TargetKeyId' => $keyId,
            'AliasName' => $alias,
        ]);
    }catch (KmsException $caught){
        if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
            echo "The request was rejected because the specified alias name
is not valid.";
        }
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param string $granteePrincipal
 * @param array $operations
 * @param array $grantTokens
 * @return Result
 */
public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
{
    $args = [
        'KeyId' => $keyId,
        'GranteePrincipal' => $granteePrincipal,
        'Operations' => $operations,
    ];
};
```

```
        if($grantTokens){
            $args['GrantTokens'] = $grantTokens;
        }
        try{
            return $this->client->createGrant($args);
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
                echo "The request was rejected because the specified grant token
is not valid.\n";
            }
            throw $caught;
        }
    }

    /**
     * @param string $keyId
     * @return array
     */
    public function describeKey(string $keyId)
    {
        try {
            $result = $this->client->describeKey([
                "KeyId" => $keyId,
            ]);
            return $result['KeyMetadata'];
        }catch(KmsException $caught){
            if($caught->getAwsErrorMessage() == "NotFoundException"){
                echo "The request was rejected because the specified entity or
resource could not be found.\n";
            }
            throw $caught;
        }
    }

    /**
     * @param string $keyId
     * @return void
     */
    public function disableKey(string $keyId)
    {

```

```
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        } catch (KmsException $caught){
            echo "There was a problem disabling the key: {"$caught->getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
}

/**
 * @param string $keyId
 * @return void
 */
public function enableKey(string $keyId)
{
    try {
        $this->client->enableKey([
            'KeyId' => $keyId,
        ]);
    } catch (KmsException $caught){
        if($caught->getAwsErrorMessage() == "NotFoundException"){
            echo "The request was rejected because the specified entity or
resource could not be found.\n";
        }
        throw $caught;
    }
}

/**
 * @return array
 */
public function listKeys()
{
    try {
        $contents = [];
        $paginator = $this->client->getPaginator("ListKeys");
        foreach($paginator as $result){
            foreach ($result['Content'] as $object) {
```

```

        $contents[] = $object;
    }
}
return $contents;
}catch(KmsException $caught){
    echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
    throw $caught;
}
}

/**
 * @param string $keyId
 * @return Result
 */
public function listGrants(string $keyId)
{
    try{
        return $this->client->listGrants([
            'KeyId' => $keyId,
        ]);
    }catch(KmsException $caught){
        if($caught->getAwsErrorMessage() == "NotFoundException"){
            echo "    The request was rejected because the specified entity
or resource could not be found.\n";
        }
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @return Result
 */
public function getKeyPolicy(string $keyId)
{
    try {
        return $this->client->getKeyPolicy([
            'KeyId' => $keyId,
        ]);
    }catch(KmsException $caught){

```

```
        echo "There was a problem getting the key policy: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}

/**
 * @param string $grantId
 * @param string $keyId
 * @return void
 */
public function revokeGrant(string $grantId, string $keyId)
{
    try{
        $this->client->revokeGrant([
            'GrantId' => $grantId,
            'KeyId' => $keyId,
        ]);
    }catch(KmsException $caught){
        echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param int $pendingWindowInDays
 * @return void
 */
public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
7)
{
    try {
        $this->client->scheduleKeyDeletion([
            'KeyId' => $keyId,
            'PendingWindowInDays' => $pendingWindowInDays,
        ]);
    }catch(KmsException $caught){
        echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
    }
}
```

```
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param array $tags
 * @return void
 */
public function tagResource(string $keyId, array $tags)
{
    try {
        $this->client->tagResource([
            'KeyId' => $keyId,
            'Tags' => $tags,
        ]);
    } catch (KmsException $caught) {
        echo "There was a problem applying the tag(s): {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param string $message
 * @param string $algorithm
 * @return Result
 */
public function sign(string $keyId, string $message, string $algorithm)
{
    try {
        return $this->client->sign([
            'KeyId' => $keyId,
            'Message' => $message,
            'SigningAlgorithm' => $algorithm,
        ]);
    } catch (KmsException $caught) {
        echo "There was a problem signing the data: {$caught-
>getAwsErrorMessage()}\n";
    }
}
```

```
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param int $rotationPeriodInDays
 * @return void
 */
public function enableKeyRotation(string $keyId, int $rotationPeriodInDays =
365)
{
    try{
        $this->client->enableKeyRotation([
            'KeyId' => $keyId,
            'RotationPeriodInDays' => $rotationPeriodInDays,
        ]);
    }catch(KmsException $caught){
        if($caught->getAwsErrorMessage() == "NotFoundException"){
            echo "The request was rejected because the specified entity or
resource could not be found.\n";
        }
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param string $policy
 * @return void
 */
public function putKeyPolicy(string $keyId, string $policy)
{
    try {
        $this->client->putKeyPolicy([
            'KeyId' => $keyId,
            'Policy' => $policy,
        ]);
    }catch(KmsException $caught){
```



```
        echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}

/**
 * @param string $aliasName
 * @return void
 */
public function deleteAlias(string $aliasName)
{
    try {
        $this->client->deleteAlias([
            'AliasName' => $aliasName,
        ]);
    } catch (KmsException $caught){
        echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}

/**
 * @param string $keyId
 * @param string $message
 * @param string $signature
 * @param string $signingAlgorithm
 * @return bool
 */
public function verify(string $keyId, string $message, string $signature,
string $signingAlgorithm)
{
    try {
        $result = $this->client->verify([
            'KeyId' => $keyId,
            'Message' => $message,
            'Signature' => $signature,
            'SigningAlgorithm' => $signingAlgorithm,
        ]);
    }
```

```
        return $result['SignatureValid'];
    }catch(KmsException $caught){
        echo "There was a problem verifying the signature: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}
}
```

- APIの詳細については、『AWS SDK for PHP API リファレンス』の以下のトピックを参照してください。

- [CreateAlias](#)
- [CreateGrant](#)
- [CreateKey](#)
- [Decrypt](#)
- [DescribeKey](#)
- [DisableKey](#)
- [EnableKey](#)
- [暗号化](#)
- [GetKeyPolicy](#)
- [ListAliases](#)
- [ListGrants](#)
- [ListKeys](#)
- [RevokeGrant](#)
- [ScheduleKeyDeletion](#)
- [Sign](#)
- [TagResource](#)

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KMSScenario:
    """Runs an interactive scenario that shows how to get started with KMS."""

    def __init__(
        self,
        key_manager: KeyManager,
        key_encryption: KeyEncrypt,
        alias_manager: AliasManager,
        grant_manager: GrantManager,
        key_policy: KeyPolicy,
    ):
        self.key_manager = key_manager
        self.key_encryption = key_encryption
        self.alias_manager = alias_manager
        self.grant_manager = grant_manager
        self.key_policy = key_policy
        self.key_id = ""
        self.alias_name = ""
        self.asymmetric_key_id = ""

    def kms_scenario(self):
        key_description = "Created by the AWS KMS API"

        print(DASHES)
        print(
            """
Welcome to the AWS Key Management SDK Basics scenario.

This program demonstrates how to interact with AWS Key Management using the AWS
SDK for Python (Boto3).
The AWS Key Management Service (KMS) is a secure and highly available service
that allows you to create
```

and manage AWS KMS keys and control their use across a wide range of AWS services and applications.

KMS provides a centralized and unified approach to managing encryption keys, making it easier to meet your data protection and regulatory compliance requirements.

This Basics scenario creates two key types:

- A symmetric encryption key is used to encrypt and decrypt data.
- An asymmetric key used to digitally sign data.

Let's get started...

```
    """
    )
    q.ask("Press Enter to continue...")

    print(DASHES)
    print(f"1. Create a symmetric KMS key\n")
    print(
        f"First, the program will creates a symmetric KMS key that you can
used to encrypt and decrypt data."
    )
    q.ask("Press Enter to continue...")
    self.key_id = self.key_manager.create_key(key_description)["KeyId"]
    print(f"A symmetric key was successfully created {self.key_id}.")
    q.ask("Press Enter to continue...")
    print(DASHES)
    print(
        """
```

2. Enable a KMS key

By default, when the SDK creates an AWS key, it is enabled. The next bit of code checks to determine if the key is enabled.

```
    """
    )
    q.ask("Press Enter to continue...")
    is_enabled = self.is_key_enabled(self.key_id)
    print(f"Is the key enabled? {is_enabled}")
    if not is_enabled:
        self.key_manager.enable_key(self.key_id)
    q.ask("Press Enter to continue...")
    print(DASHES)
    print(f"3. Encrypt data using the symmetric KMS key")
```

```

    plain_text = "Hello, AWS KMS!"
    print(
        f"""

```

One of the main uses of symmetric keys is to encrypt and decrypt data.

Next, the code encrypts the string "{plain_text}" with the SYMMETRIC_DEFAULT encryption algorithm.

```

        """
    )
    q.ask("Press Enter to continue...")
    encrypted_text = self.key_encryption.encrypt(self.key_id, plain_text)
    print(DASHES)
    print(f"4. Create an alias")
    print(
        """

```

Now, the program will create an alias for the KMS key. An alias is a friendly name that you

can associate with a KMS key. The alias name should be prefixed with 'alias/'.

```

        """
    )
    alias_name = q.ask("Enter an alias name: ", q.non_empty)
    self.alias_manager.create_alias(self.key_id, alias_name)
    print(f"{alias_name} was successfully created.")
    self.alias_name = alias_name
    print(DASHES)
    print(f"5. List all of your aliases")
    q.ask("Press Enter to continue...")
    self.alias_manager.list_aliases(10)
    q.ask("Press Enter to continue...")
    print(DASHES)
    print(f"6. Enable automatic rotation of the KMS key")
    print(
        """

```

By default, when the SDK enables automatic rotation of a KMS key, KMS rotates the key material of the KMS key one year (approximately 365 days) from the enable date and every year thereafter.

```

        """
    )
    q.ask("Press Enter to continue...")
    self.key_manager.enable_key_rotation(self.key_id)
    print(DASHES)
    print(f"Key rotation has been enabled for key with id {self.key_id}")
    print(

```

```

    """
7. Create a grant

```

A grant is a policy instrument that allows Amazon Web Services principals to use KMS keys.

It also can allow them to view a KMS key (DescribeKey) and create and manage grants.

When authorizing access to a KMS key, grants are considered along with key policies and IAM policies.

```

    """
    )
    print(
        """

To create a grant you must specify a account_id. To specify the grantee
account_id, use the Amazon Resource Name (ARN)
of an AWS account_id. Valid principals include AWS accounts, IAM users, IAM
roles, federated users,
and assumed role users.
    """
    )
    account_id = q.ask(
        "Enter an account_id, or press enter to skip creating a grant... "
    )
    grant = None
    if account_id != "":
        grant = self.grant_manager.create_grant(
            self.key_id,
            account_id,
            [
                "Encrypt",
                "Decrypt",
                "DescribeKey",
            ],
        )
        print(f"Grant created successfully with ID: {grant['GrantId']}")

    q.ask("Press Enter to continue...")
    print(DASHES)
    print(DASHES)
    print(f"8. List grants for the KMS key")
    q.ask("Press Enter to continue...")
    self.grant_manager.list_grants(self.key_id)
    q.ask("Press Enter to continue...")
    print(DASHES)

```

```

    print(f"9. Revoke the grant")
    print(
        """

```

The revocation of a grant immediately removes the permissions and access that the grant had provided.

This means that any `account_id` (user, role, or service) that was granted access to perform specific

KMS operations on a KMS key will no longer be able to perform those operations.

```

        """
    )
    q.ask("Press Enter to continue...")

    if grant is not None:
        self.grant_manager.revoke_grant(self.key_id, grant["GrantId"])
        print(f"Grant ID: {grant['GrantId']} was successfully revoked!")

```

```

    q.ask("Press Enter to continue...")
    print(DASHES)
    print(f"10. Decrypt the data\n")
    print(
        """

```

Lets decrypt the data that was encrypted in an early step.

The code uses the same key to decrypt the string that we encrypted earlier in the program.

```

        """
    )
    q.ask("Press Enter to continue...")
    decrypted_data = self.key_encryption.decrypt(self.key_id, encrypted_text)
    print(f"Data decrypted successfully for key ID: {self.key_id}")
    print(f"Decrypted data: {decrypted_data}")

```

```

    q.ask("Press Enter to continue...")
    print(DASHES)
    print(f"11. Replace a key policy\n")
    print(
        """

```

A key policy is a resource policy for a KMS key. Key policies are the primary way to control

access to KMS keys. Every KMS key must have exactly one key policy. The statements in the key policy

determine who has permission to use the KMS key and how they can use it.

You can also use IAM policies and grants to control access to the KMS key, but every KMS key

must have a key policy.

By default, when you create a key by using the SDK, a policy is created that gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": "arn:aws:iam::000000000000:root"},
    "Action": "kms:*",
    "Resource": "*"
  }]
}
```

```
"""
)
account_id = q.ask("Enter your account ID or press enter to skip: ")
if account_id != "":
    policy = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"AWS": f"arn:aws:iam::{account_id}:root"},
                "Action": "kms:*",
                "Resource": "*",
            }
        ],
    }

    self.key_policy.set_new_policy(self.key_id, policy)
    print("Key policy replacement succeeded.")
    q.ask("Press Enter to continue...")
else:
    print("Skipping replacing the key policy.")

print(DASHES)
print(f"12. Get the key policy\n")
print(
    f"The next bit of code that runs gets the key policy to make sure it
exists."
)
q.ask("Press Enter to continue...")
policy = self.key_policy.get_policy(self.key_id)
```



```

print(f"The key policy is: {policy}")

q.ask("Press Enter to continue...")
print(DASHES)
print(f"13. Create an asymmetric KMS key and sign your data\n")
print(
    """
    Signing your data with an AWS key can provide several benefits that make
it an attractive option
    for your data signing needs. By using an AWS KMS key, you can leverage
the
    security controls and compliance features provided by AWS,
    which can help you meet various regulatory requirements and enhance the
overall security posture
    of your organization.
    """
)
q.ask("Press Enter to continue...")
print(f"Sign and verify data operation succeeded.")
self.asymmetric_key_id = self.key_manager.create_asymmetric_key()
message = "Here is the message that will be digitally signed"
signature = self.key_encryption.sign(self.asymmetric_key_id, message)
if self.key_encryption.verify(self.asymmetric_key_id, message,
signature):
    print("Signature verification succeeded.")
else:
    print("Signature verification failed.")

q.ask("Press Enter to continue...")
print(DASHES)
print(f"14. Tag your symmetric KMS Key\n")
print(
    """
    By using tags, you can improve the overall management, security, and
governance of your
    KMS keys, making it easier to organize, track, and control access to your
encrypted data within
    your AWS environment
    """
)
q.ask("Press Enter to continue...")
self.key_manager.tag_resource(self.key_id, "Environment", "Production")
self.clean_up()

```

```
def is_key_enabled(self, key_id: str) -> bool:
    """
    Check if the key is enabled or not.

    :param key_id: The key to check.
    :return: True if the key is enabled, otherwise False.
    """
    response = self.key_manager.describe_key(key_id)
    return response["Enabled"] is True
```

```
def clean_up(self):
    """
    Delete resources created by this scenario.
    """
    if self.alias_name != "":
        print(f"Deleting the alias {self.alias_name}.")
        self.alias_manager.delete_alias(self.alias_name)
    window = 7 # The window in days for a scheduled deletion.
    if self.key_id != "":
        print(
            """
```

Warning:

Deleting a KMS key is a destructive and potentially dangerous operation. When a KMS key is deleted, all data that was encrypted under the KMS key is unrecoverable.

```
        """
    )
    if q.ask(
        f"Do you want to delete the key with ID {self.key_id} (y/n)?",
        q.is_yesno,
    ):
        print(
            f"The key {self.key_id} will be deleted with a window of {window} days. You can cancel the deletion before"
        )
        print("the window expires.")
        self.key_manager.delete_key(self.key_id, window)
        self.key_id = ""

    if self.asymmetric_key_id != "":
        if q.ask(
            f"Do you want to delete the asymmetric key with ID {self.asymmetric_key_id} (y/n)?",
            q.is_yesno,
```

```

        ):
            print(
                f"The key {self.asymmetric_key_id} will be deleted with a
window of {window} days. You can cancel the deletion before"
            )
            print("the window expires.")
            self.key_manager.delete_key(self.asymmetric_key_id, window)
            self.asymmetric_key_id = ""

if __name__ == "__main__":
    kms_scenario = None
    try:
        kms_client = boto3.client("kms")
        a_key_manager = KeyManager(kms_client)
        a_key_encrypt = KeyEncrypt(kms_client)
        an_alias_manager = AliasManager(kms_client)
        a_grant_manager = GrantManager(kms_client)
        a_key_policy = KeyPolicy(kms_client)
        kms_scenario = KMSScenario(
            key_manager=a_key_manager,
            key_encryption=a_key_encrypt,
            alias_manager=an_alias_manager,
            grant_manager=a_grant_manager,
            key_policy=a_key_policy,
        )
        kms_scenario.kms_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo!")
        if kms_scenario is not None:
            kms_scenario.clean_up()

```

KMS キー管理のラッパークラスとメソッド。

```

class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":

```

```
"""
Creates a KeyManager instance with a default KMS client.

:return: An instance of KeyManager initialized with the default KMS
client.
"""
kms_client = boto3.client("kms")
return cls(kms_client)

def create_key(self, key_description: str) -> dict[str, any]:
    """
    Creates a key with a user-provided description.

    :param key_description: A description for the key.
    :return: The key ID.
    """
    try:
        key = self.kms_client.create_key(Description=key_description)
        ["KeyMetadata"]
        self.created_keys.append(key)
        return key
    except ClientError as err:
        logging.error(
            "Couldn't create your key. Here's why: %s",
            err.response["Error"]["Message"],
        )
        raise

def describe_key(self, key_id: str) -> dict[str, any]:
    """
    Describes a key.

    :param key_id: The ARN or ID of the key to describe.
    :return: Information about the key.
    """
    try:
        key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
        return key
    except ClientError as err:
        logging.error(
            "Couldn't get key '%s'. Here's why: %s",

```

```
        key_id,
        err.response["Error"]["Message"],
    )
    raise

def enable_key_rotation(self, key_id: str) -> None:
    """
    Enables rotation for a key.

    :param key_id: The ARN or ID of the key to enable rotation for.
    """
    try:
        self.kms_client.enable_key_rotation(KeyId=key_id)
    except ClientError as err:
        logging.error(
            "Couldn't enable rotation for key '%s'. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )
        raise

def create_asymmetric_key(self) -> str:
    """
    Creates an asymmetric key in AWS KMS for signing messages.

    :return: The ID of the created key.
    """
    try:
        key = self.kms_client.create_key(
            KeySpec="RSA_2048", KeyUsage="SIGN_VERIFY", Origin="AWS_KMS"
        )["KeyMetadata"]
        self.created_keys.append(key)
        return key["KeyId"]
    except ClientError as err:
        logger.error(
            "Couldn't create your key. Here's why: %s",
            err.response["Error"]["Message"],
        )
        raise

def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
```

```

"""
Add or edit tags on a customer managed key.

:param key_id: The ARN or ID of the key to enable rotation for.
:param tag_key: Key for the tag.
:param tag_value: Value for the tag.
"""
try:
    self.kms_client.tag_resource(
        KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
    )
except ClientError as err:
    logging.error(
        "Couldn't add a tag for the key '%s'. Here's why: %s",
        key_id,
        err.response["Error"]["Message"],
    )
    raise

def delete_key(self, key_id: str, window: int) -> None:
    """
    Deletes a list of keys.

    Warning:
    Deleting a KMS key is a destructive and potentially dangerous operation.
    When a KMS key is deleted,
    all data that was encrypted under the KMS key is unrecoverable.

    :param key_id: The ARN or ID of the key to delete.
    :param window: The waiting period, in days, before the KMS key is
    deleted.
    """
    try:
        self.kms_client.schedule_key_deletion(
            KeyId=key_id, PendingWindowInDays=window
        )
    except ClientError as err:
        logging.error(
            "Couldn't delete key %s. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )

```

```
raise
```

KMS キーエイリアスのラッパークラスとメソッド。

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
                    key_id
                )
            else:
                logger.error(
                    "Couldn't encrypt text. Here's why: %s",
                    err.response["Error"]["Message"],
                )
```

```

        raise

def list_aliases(self, page_size: int) -> None:
    """
    Lists aliases for the current account.
    :param page_size: The number of aliases to list per page.
    """
    try:
        alias_paginator = self.kms_client.get_paginator("list_aliases")
        for alias_page in alias_paginator.paginate(
            PaginationConfig={"PageSize": page_size}
        ):
            print(f"Here are {page_size} aliases:")
            pprint(alias_page["Aliases"])
            if alias_page["Truncated"]:
                answer = input(
                    f"Do you want to see the next {page_size} aliases (y/n)?
                "
                )
                if answer.lower() != "y":
                    break
            else:
                print("That's all your aliases!")
    except ClientError as err:
        logging.error(
            "Couldn't list your aliases. Here's why: %s",
            err.response["Error"]["Message"],
        )
        raise

def delete_alias(self, alias: str) -> None:
    """
    Deletes an alias.

    :param alias: The alias to delete.
    """
    try:
        self.kms_client.delete_alias(AliasName=alias)
    except ClientError as err:
        logger.error(
            "Couldn't delete alias %s. Here's why: %s",
            alias,

```



```
        err.response["Error"]["Message"],
    )
    raise
```

KMS キー暗号化のラッパークラスとメソッド。

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def encrypt(self, key_id: str, text: str) -> str:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
{response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DisabledException":
```

```
        logger.error(
            "Could not encrypt because the key %s is disabled.", key_id
        )
    else:
        logger.error(
            "Couldn't encrypt text. Here's why: %s",
            err.response["Error"]["Message"],
        )
    raise

def decrypt(self, key_id: str, cipher_text: str) -> bytes:
    """
    Decrypts text previously encrypted with a key.

    :param key_id: The ARN or ID of the key used to decrypt the data.
    :param cipher_text: The encrypted text to decrypt.
    :return: The decrypted text.
    """
    try:
        return self.kms_client.decrypt(KeyId=key_id,
            CiphertextBlob=cipher_text)[
            "Plaintext"
        ]
    except ClientError as err:
        logger.error(
            "Couldn't decrypt your ciphertext. Here's why: %s",
            err.response["Error"]["Message"],
        )
    raise

def sign(self, key_id: str, message: str) -> str:
    """
    Signs a message with a key.

    :param key_id: The ARN or ID of the key to use for signing.
    :param message: The message to sign.
    :return: The signature of the message.
    """
    try:
        return self.kms_client.sign(
            KeyId=key_id,
            Message=message.encode(),

```

```
        SigningAlgorithm="RSASSA_PSS_SHA_256",
    )["Signature"]
except ClientError as err:
    logger.error(
        "Couldn't sign your message. Here's why: %s",
        err.response["Error"]["Message"],
    )
    raise

def verify(self, key_id: str, message: str, signature: str) -> bool:
    """
    Verifies a signature against a message.

    :param key_id: The ARN or ID of the key used to sign the message.
    :param message: The message to verify.
    :param signature: The signature to verify.
    :return: True when the signature matches the message, otherwise False.
    """
    try:
        response = self.kms_client.verify(
            KeyId=key_id,
            Message=message.encode(),
            Signature=signature,
            SigningAlgorithm="RSASSA_PSS_SHA_256",
        )
        valid = response["SignatureValid"]
        print(f"The signature is {'valid' if valid else 'invalid'}.")
        return valid
    except ClientError as err:
        if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
            print("The signature is not valid.")
        else:
            logger.error(
                "Couldn't verify your signature. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

KMS キー許可のラッパークラスとメソッド。

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
encryption key.

        :param key_id: The ARN or ID of the key.
        :param principal: The principal to grant permission to.
        :param operations: The operations to grant permission for.
        :return: The grant that is created.
        """
        try:
            return self.kms_client.create_grant(
                KeyId=key_id,
                GranteePrincipal=principal,
                Operations=operations,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create a grant on key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
```

```
def list_grants(self, key_id):
    """
    Lists grants for a key.

    :param key_id: The ARN or ID of the key to query.
    :return: The grants for the key.
    """
    try:
        paginator = self.kms_client.get_paginator("list_grants")
        grants = []
        page_iterator = paginator.paginate(KeyId=key_id)
        for page in page_iterator:
            grants.extend(page["Grants"])

        print(f"Grants for key {key_id}:")
        pprint(grants)
        return grants
    except ClientError as err:
        logger.error(
            "Couldn't list grants for key %s. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )
        raise

def revoke_grant(self, key_id: str, grant_id: str) -> None:
    """
    Revokes a grant so that it can no longer be used.

    :param key_id: The ARN or ID of the key associated with the grant.
    :param grant_id: The ID of the grant to revoke.
    """
    try:
        self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
    except ClientError as err:
        logger.error(
            "Couldn't revoke grant %s. Here's why: %s",
            grant_id,
            err.response["Error"]["Message"],
        )
        raise
```

KMS キーポリシーのラッパークラスとメソッド。

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def set_new_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """
        Sets the policy of a key. Setting a policy entirely overwrites the
existing
        policy, so care is taken to add a statement to the existing list of
statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: A new key policy. The key policy must allow the calling
principal to make a subsequent
            PutKeyPolicy request on the KMS key. This reduces the risk
that the KMS key becomes unmanageable
        """

        try:
            self.kms_client.put_key_policy(KeyId=key_id,
Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %s. Here's why %s",
                key_id,
                err.response["Error"]["Message"],
```

```
    )
    raise

def get_policy(self, key_id: str) -> dict[str, str]:
    """
    Gets the policy of a key.

    :param key_id: The ARN or ID of the key to query.
    :return: The key policy as a dict.
    """
    if key_id != "":
        try:
            response = self.kms_client.get_key_policy(
                KeyId=key_id,
            )
            policy = json.loads(response["Policy"])
        except ClientError as err:
            logger.error(
                "Couldn't get policy for key %s. Here's why: %s",
                key_id,
                err.response["Error"]["Message"],
            )
            raise
        else:
            pprint(policy)
            return policy
    else:
        print("Skipping get policy demo.")
```

- APIの詳細については、『AWS SDK for Python (Boto3) API リファレンス』の以下のトピックを参照してください。
 - [CreateAlias](#)
 - [CreateGrant](#)
 - [CreateKey](#)
 - [Decrypt](#)
 - [DescribeKey](#)
 - [DisableKey](#)

- [EnableKey](#)
- [暗号化](#)
- [GetKeyPolicy](#)
- [ListAliases](#)
- [ListGrants](#)
- [ListKeys](#)
- [RevokeGrant](#)
- [ScheduleKeyDeletion](#)
- [Sign](#)
- [TagResource](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs AWS KMS を使用するためのアクション

次のコード例は、AWS SDKs で個々の AWS KMS アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

これらの抜粋は AWS KMS API を呼び出し、コンテキスト内で実行する必要がある大規模なプログラムからのコード抜粋です。アクションは [AWS SDKs AWS KMS を使用するシナリオ](#) のコンテキスト内で確認できます。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「[AWS Key Management Service API リファレンス](#)」を参照してください。

例

- [AWS SDK または CLI CreateAliasで を使用する](#)
- [AWS SDK または CLI CreateGrantで を使用する](#)
- [AWS SDK または CLI CreateKeyで を使用する](#)
- [AWS SDK または CLI Decryptで を使用する](#)
- [AWS SDK または CLI DeleteAliasで を使用する](#)

- [AWS SDK または CLI DescribeKey で 使用する](#)
- [AWS SDK または CLI DisableKey で 使用する](#)
- [AWS SDK または CLI EnableKey で 使用する](#)
- [AWS SDK または CLI EnableKeyRotation で 使用する](#)
- [AWS SDK または CLI Encrypt で 使用する](#)
- [AWS SDK または CLI GenerateDataKey で 使用する](#)
- [AWS SDK または CLI GenerateDataKeyWithoutPlaintext で 使用する](#)
- [AWS SDK または CLI GenerateRandom で 使用する](#)
- [AWS SDK または CLI GetKeyPolicy で 使用する](#)
- [AWS SDK または CLI ListAliases で 使用する](#)
- [AWS SDK または CLI ListGrants で 使用する](#)
- [AWS SDK または CLI ListKeyPolicies で 使用する](#)
- [AWS SDK または CLI ListKeys で 使用する](#)
- [AWS SDK または CLI PutKeyPolicy で 使用する](#)
- [AWS SDK または CLI ReEncrypt で 使用する](#)
- [AWS SDK または CLI RetireGrant で 使用する](#)
- [AWS SDK または CLI RevokeGrant で 使用する](#)
- [AWS SDK または CLI ScheduleKeyDeletion で 使用する](#)
- [AWS SDK または CLI Sign で 使用する](#)
- [AWS SDK または CLI TagResource で 使用する](#)
- [AWS SDK または CLI UpdateAlias で 使用する](#)
- [AWS SDK または CLI Verify で 使用する](#)

AWS SDK または CLI **CreateAlias** で 使用する

次のサンプルコードは、CreateAlias を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Creates an alias for an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";

        // The value supplied as the TargetKeyId can be either
        // the key ID or key Amazon Resource Name (ARN) of the
        // AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new CreateAliasRequest
        {
            AliasName = aliasName,
            TargetKeyId = keyId,
        };

        var response = await client.CreateAliasAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
        {
            Console.WriteLine($"Alias, {aliasName}, successfully created.");
        }
        else
        {
            Console.WriteLine($"Could not create alias.");
        }
    }
}
```

- API の詳細については、AWS SDK for .NET API リファレンスの[CreateAlias](#)を参照してください。

CLI

AWS CLI

KMS キーのエイリアスを作成するには

次の create-alias コマンドは、キー ID 1234abcd-12ab-34cd-56ef-1234567890ab で識別される KMS キーの example-alias という名前のエイリアスを作成します。

エイリアス名は alias/ で始める必要があります。で始まるエイリアス名は使用しないでくださいalias/aws。これらは 用に予約されています AWS。

```
aws kms create-alias \  
  --alias-name alias/example-alias \  
  --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

このコマンドは出力を返しません。新しいエイリアスを確認するには、list-aliases コマンドを使用します。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[エイリアスの使用](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[CreateAlias](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Creates a custom alias for the specified target key asynchronously.
 *
 * @param targetKeyId the ID of the target key for the alias
 * @param aliasName the name of the alias to create
 * @return a {@link CompletableFuture} that completes when the alias creation
 * operation is finished
 */
public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId,
String aliasName) {
    CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
        .aliasName(aliasName)
        .targetKeyId(targetKeyId)
        .build();

    CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
    responseFuture.whenComplete((response, exception) -> {
        if (exception == null) {
            logger.info("{} was successfully created.", aliasName);
        } else {
            if (exception instanceof ResourceExistsException) {
                logger.info("Alias [{}] already exists. Moving on...",
aliasName);
            } else if (exception instanceof KmsException kmsEx) {
                throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
            } else {
                throw new RuntimeException("An unexpected error occurred
while creating alias: " + exception.getMessage(), exception);
            }
        }
    })
}
```

```
});  
  
return responseFuture.thenApply(response -> null);  
}
```

- API の詳細については、AWS SDK for Java 2.x API リファレンスの「[CreateAlias](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createCustomAlias(  
    targetKeyIdVal: String?,  
    aliasNameVal: String?,  
) {  
    val request =  
        CreateAliasRequest {  
            aliasName = aliasNameVal  
            targetKeyId = targetKeyIdVal  
        }  
  
    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->  
        kmsClient.createAlias(request)  
        println("$aliasNameVal was successfully created")  
    }  
}
```

- API の詳細については、「AWS SDK for Kotlin API リファレンス」の「[CreateAlias](#)」を参照してください。

PHP

SDK for PHP

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param string $alias
 * @return void
 */
public function createAlias(string $keyId, string $alias)
{
    try{
        $this->client->createAlias([
            'TargetKeyId' => $keyId,
            'AliasName' => $alias,
        ]);
    }catch (KmsException $caught){
        if($caught->getAwsErrorMessage() == "InvalidAliasNameException"){
            echo "The request was rejected because the specified alias name
is not valid.";
        }
        throw $caught;
    }
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの[CreateAlias](#)を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def create_alias(self, key_id: str, alias: str) -> None:
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :param alias: The alias to assign to the key.
        """
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExistsException":
                logger.error(
                    "Could not create the alias %s because it already exists.",
key_id
                )
```

```
else:
    logger.error(
        "Couldn't encrypt text. Here's why: %s",
        err.response["Error"]["Message"],
    )
    raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[CreateAlias](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateGrant**で を使用する

次のサンプルコードは、CreateGrant を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
public static async Task Main()
{
    var client = new AmazonKeyManagementServiceClient();
```



```

// The identity that is given permission to perform the operations
// specified in the grant.
var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

// The identifier of the AWS KMS key to which the grant applies. You
// can use the key ID or the Amazon Resource Name (ARN) of the KMS
key.
var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";

var request = new CreateGrantRequest
{
    GranteePrincipal = grantee,
    KeyId = keyId,

    // A list of operations that the grant allows.
    Operations = new List<string>
    {
        "Encrypt",
        "Decrypt",
    },
};

var response = await client.CreateGrantAsync(request);

string grantId = response.GrantId; // The unique identifier of the
grant.
string grantToken = response.GrantToken; // The grant token.

Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
}
}

```

- APIの詳細については、AWS SDK for .NET API リファレンスの[CreateGrant](#)を参照してください。

CLI

AWS CLI

権限を作成するには

次の `create-grant` の例は、`exampleUser` ユーザーが `1234abcd-12ab-34cd-56ef-1234567890ab` の KMS キーで `decrypt` コマンドを使用する権限を作成します。廃止するプリンシパルは `adminRole` ロールです。この権限は `EncryptionContextSubset` 権限の制約を使用して `"Department": "IT"` キー値ペアを含む `decrypt` リクエストでの暗号化のみを許可します。

```
aws kms create-grant \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --grantee-principal arn:aws:iam::123456789012:user/exampleUser \  
  --operations Decrypt \  
  --constraints EncryptionContextSubset={Department=IT} \  
  --retiring-principal arn:aws:iam::123456789012:role/adminRole
```

出力:

```
{  
  "GrantId":  
    "1a2b3c4d2f5e69f440bae30eaec9570bb1fb7358824f9ddfa1aa5a0dab1a59b2",  
  "GrantToken": "<grant token here>"  
}
```

権限に関する詳細情報を表示するには、`list-grants` コマンドを使用します。

詳細については、AWS 「Key Management Service デベロッパーガイド」の [「KMS AWS で の Grant」](#) を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの [「CreateGrant」](#) を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。 [AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
```

```

    * Grants permissions to a specified principal on a customer master key (CMK)
    asynchronously.
    *
    * @param keyId          The unique identifier for the customer master key
    (CMK) that the grant applies to.
    * @param granteePrincipal The principal that is given permission to perform
    the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID
    of the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
    process.
    */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
    granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
        getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
                response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
                    kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
                    ex.getMessage(), ex);
                }
            }
        });
    }

```

```
        return responseFuture.thenApply(CreateGrantResponse::grantId);  
    }
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateGrant](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createNewGrant(  
    keyIdVal: String?,  
    granteePrincipalVal: String?,  
    operation: String,  
): String? {  
    val operation0b = GrantOperation.fromValue(operation)  
    val grantOperationList = ArrayList<GrantOperation>()  
    grantOperationList.add(operation0b)  
  
    val request =  
        CreateGrantRequest {  
            keyId = keyIdVal  
            granteePrincipal = granteePrincipalVal  
            operations = grantOperationList  
        }  
  
    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.createGrant(request)  
        return response.grantId  
    }  
}
```

- API の詳細については、[AWS SDK for Kotlin API リファレンス](#)の「CreateGrant」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param string $granteePrincipal
 * @param array $operations
 * @param array $grantTokens
 * @return Result
 */
public function createGrant(string $keyId, string $granteePrincipal, array
$operations, array $grantTokens = [])
{
    $args = [
        'KeyId' => $keyId,
        'GranteePrincipal' => $granteePrincipal,
        'Operations' => $operations,
    ];
    if($grantTokens){
        $args['GrantTokens'] = $grantTokens;
    }
    try{
        return $this->client->createGrant($args);
    }catch(KmsException $caught){
        if($caught->getAwsErrorMessage() == "InvalidGrantTokenException"){
            echo "The request was rejected because the specified grant token
is not valid.\n";
        }
        throw $caught;
    }
}
```

```
}
```

- API の詳細については、AWS SDK for PHP API リファレンスの[CreateGrant](#)を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def create_grant(
        self, key_id: str, principal: str, operations: [str]
    ) -> dict[str, str]:
        """
        Creates a grant for a key that lets a principal generate a symmetric data
encryption key.

        :param key_id: The ARN or ID of the key.
```

```
:param principal: The principal to grant permission to.
:param operations: The operations to grant permission for.
:return: The grant that is created.
"""
try:
    return self.kms_client.create_grant(
        KeyId=key_id,
        GranteePrincipal=principal,
        Operations=operations,
    )
except ClientError as err:
    logger.error(
        "Couldn't create a grant on key %s. Here's why: %s",
        key_id,
        err.response["Error"]["Message"],
    )
    raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[CreateGrant](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateKey**で を使用する

次のサンプルコードは、CreateKey を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)
- [テーブルの暗号化を操作する](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Shows how to create a new AWS Key Management Service (AWS KMS)
/// key.
/// </summary>
public class CreateKey
{
    public static async Task Main()
    {
        // Note that if you need to create a Key in an AWS Region
        // other than the Region defined for the default user, you need to
        // pass the Region to the client constructor.
        var client = new AmazonKeyManagementServiceClient();

        // The call to CreateKeyAsync will create a symmetrical AWS KMS
        // key. For more information about symmetrical and asymmetrical
        // keys, see:
        //
        // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
        var response = await client.CreateKeyAsync(new CreateKeyRequest());

        // The KeyMetadata object contains information about the new AWS KMS
key.
        KeyMetadata keyMetadata = response.KeyMetadata;

        if (keyMetadata is not null)
        {
```



```
        Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
created.");
    }
    else
    {
        Console.WriteLine("Could not create KMS Key.");
    }
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[CreateKey](#)」を参照してください。

CLI

AWS CLI

例 1: KMS でカスタマーマネージド KMS AWS キーを作成するには

次の create-key の例は、対称暗号化 KMS キーを作成します。

基本的な KMS キー (対称暗号化キー) を作成するには、パラメータを指定する必要はありません。対称暗号化キーは、これらのパラメータのデフォルト値によって作成されます。

このコマンドはキーポリシーを指定しないため、KMS キーにはプログラムで作成された KMS キーの[デフォルトキーポリシー](#)が適用されます。キーポリシーを表示するには、get-key-policy コマンドを使用します。キーポリシーを変更するには、put-key-policy コマンドを使用します。

```
aws kms create-key
```

この create-key コマンドは、新しい KMS キーのキー ID と ARN を含むキーメタデータを返します。これらの値を使用して、他の KMS オペレーションで KMS AWS キーを識別できます。出力にはタグは含まれません。KMS キーのタグを表示するには、list-resource-tags command を使用します。

出力:

```
{
```

```

    "KeyMetadata": {
      "AWSAccountId": "111122223333",
      "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "CreationDate": "2017-07-05T14:04:55-07:00",
      "CurrentKeyMaterialId":
"0b7fd7ddb6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
      "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
      "Description": "",
      "Enabled": true,
      "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "KeyManager": "CUSTOMER",
      "KeySpec": "SYMMETRIC_DEFAULT",
      "KeyState": "Enabled",
      "KeyUsage": "ENCRYPT_DECRYPT",
      "MultiRegion": false,
      "Origin": "AWS_KMS"
      "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
      ]
    }
  }
}

```

注: この create-key コマンドではエイリアスを指定できません。新しい KMS キーのエイリアスを作成するには、create-alias コマンドを使用してください。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[キーの作成](#)」を参照してください。

例 2: 暗号化と復号用の非対称 RSA KMS キーを作成するには

次の create-key の例は、暗号化と復号用の非対称 RSA キーペアを含む KMS キーを作成します。キーの作成後は、キー仕様とキーの使用方法を変更することはできません。

```

aws kms create-key \
  --key-spec RSA_4096 \
  --key-usage ENCRYPT_DECRYPT

```

出力:

```

{
  "KeyMetadata": {

```

```

    "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2021-04-05T14:04:55-07:00",
    "CustomerMasterKeySpec": "RSA_4096",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
        "RSAES_OAEP_SHA_1",
        "RSAES_OAEP_SHA_256"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "RSA_4096",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "AWS_KMS"
  }
}

```

詳細については、「Key Management Service デベロッパーガイド」の [「KMS AWS の非対称キー」](#) を参照してください。AWS

例 3: 署名と検証用に非対称楕円曲線 KMS キーを作成するには

署名と検証用に非対称楕円曲線 (ECC) キーペアを持つ非対称 KMS キーを作成します。SIGN_VERIFY は ECC KMS キーの唯一の有効な値ですが、--key-usage パラメータは必須です。キーの作成後は、キー仕様とキーの使用方法を変更することはできません。

```

aws kms create-key \
  --key-spec ECC_NIST_P521 \
  --key-usage SIGN_VERIFY

```

出力:

```

{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2019-12-02T07:48:55-07:00",

```

```

    "CustomerMasterKeySpec": "ECC_NIST_P521",
    "Description": "",
    "Enabled": true,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "ECC_NIST_P521",
    "KeyState": "Enabled",
    "KeyUsage": "SIGN_VERIFY",
    "MultiRegion": false,
    "Origin": "AWS_KMS",
    "SigningAlgorithms": [
        "ECDSA_SHA_512"
    ]
  }
}

```

詳細については、「Key Management Service デベロッパーガイド」の [「KMS AWS の非対称キー」](#) を参照してください。AWS

例 4: 署名と検証用に非対称 ML-DSA KMS キーを作成するには

この例では、署名と検証用の Module-Lattice Digital Signature Algorithm (ML-DSA) キーを作成します。SIGN_VERIFY は ML-DSA キーの唯一の有効な値ですが、key-usage パラメータは必須です。

```

aws kms create-key \
  --key-spec ML_DSA_65 \
  --key-usage SIGN_VERIFY

```

出力:

```

{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2019-12-02T07:48:55-07:00",
    "Description": "",
    "Enabled": true,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "ML_DSA_65",

```

```

    "KeyState": "Enabled",
    "KeyUsage": "SIGN_VERIFY",
    "MultiRegion": false,
    "Origin": "AWS_KMS",
    "SigningAlgorithms": [
        "ML_DSA_SHAKE_256"
    ]
}
}

```

詳細については、「Key Management Service デベロッパーガイド」の「[KMS AWS の非対称キー](#)」を参照してください。AWS

例 5: HMAC KMS キーを作成するには

次の create-key の例は、384-bit HMAC KMS キーを作成します。--key-usage パラメータの値 GENERATE_VERIFY_MAC は HMAC KMS キーの唯一の有効な値ですが、この値は必須です。

```

aws kms create-key \
  --key-spec HMAC_384 \
  --key-usage GENERATE_VERIFY_MAC

```

出力:

```

{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2022-04-05T14:04:55-07:00",
    "CustomerMasterKeySpec": "HMAC_384",
    "Description": "",
    "Enabled": true,
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "HMAC_384",
    "KeyState": "Enabled",
    "KeyUsage": "GENERATE_VERIFY_MAC",
    "MacAlgorithms": [
      "HMAC_SHA_384"
    ],
  },
}

```

```
    "MultiRegion": false,  
    "Origin": "AWS_KMS"  
  }  
}
```

詳細については、「Key Management Service デベロッパーガイド」の「[KMS AWS の HMAC キー](#)」を参照してください。AWS

例 6: マルチリージョンのプライマリ KMS キーを作成するには

次の create-key の例は、マルチリージョンのプライマリ対称暗号化キーを作成します。すべてのパラメータのデフォルト値によって対称暗号化キーが作成されるため、この KMS キーに必要なのは --multi-region パラメータだけです。AWS CLI で、ブールパラメータが true であることを示すには、パラメータ名を指定します。

```
aws kms create-key \  
  --multi-region
```

出力:

```
{  
  "KeyMetadata": {  
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/  
mrk-1234abcd12ab34cd56ef12345678990ab",  
    "AWSAccountId": "111122223333",  
    "CreationDate": "2021-09-02T016:15:21-09:00",  
    "CurrentKeyMaterialId":  
"0b7fd7ddb6c6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",  
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",  
    "Description": "",  
    "Enabled": true,  
    "EncryptionAlgorithms": [  
      "SYMMETRIC_DEFAULT"  
    ],  
    "KeyId": "mrk-1234abcd12ab34cd56ef12345678990ab",  
    "KeyManager": "CUSTOMER",  
    "KeySpec": "SYMMETRIC_DEFAULT",  
    "KeyState": "Enabled",  
    "KeyUsage": "ENCRYPT_DECRYPT",  
    "MultiRegion": true,  
    "MultiRegionConfiguration": {  
      "MultiRegionKeyType": "PRIMARY",
```

```

    "PrimaryKey": {
      "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef12345678990ab",
      "Region": "us-west-2"
    },
    "ReplicaKeys": []
  },
  "Origin": "AWS_KMS"
}

```

詳細については、「Key Management Service デベロッパーガイド」の「[KMS AWS の非対称キー](#)」を参照してください。AWS

例 7: インポートされたキーマテリアルの KMS キーを作成するには

次の create-key の例は、キーマテリアルなしで KMS キーを作成します。操作が完了すると、独自のキーマテリアルを KMS キーにインポートできます。この KMS キーを作成するには、--origin パラメータを EXTERNAL に設定します。

```

aws kms create-key \
  --origin EXTERNAL

```

出力:

```

{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CreationDate": "2019-12-02T07:48:55-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "Description": "",
    "Enabled": false,
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "PendingImport",
    "KeyUsage": "ENCRYPT_DECRYPT",
  }
}

```

```

        "MultiRegion": false,
        "Origin": "EXTERNAL"
    }
}

```

詳細については、[「Key Management Service デベロッパーガイド」の「KMS AWS キーでのキーマテリアルのインポート」](#)を参照してください。AWS

例 6: AWS CloudHSM キーストアに KMS キーを作成するには

次の `create-key` 例では、指定された AWS CloudHSM キーストアに KMS キーを作成します。オペレーションは、KMS で KMS AWS キーとそのメタデータを作成し、カスタムキーストアに関連付けられた AWS CloudHSM クラスターにキーマテリアルを作成します。--`custom-key-store-id` および --`origin` パラメータが必要です。

```

aws kms create-key \
  --origin AWS_CLOUDHSM \
  --custom-key-store-id cks-1234567890abcdef0

```

出力:

```

{
  "KeyMetadata": {
    "Arn": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "AWSAccountId": "111122223333",
    "CloudHsmClusterId": "cluster-1a23b4cdefg",
    "CreationDate": "2019-12-02T07:48:55-07:00",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "CustomKeyId": "cks-1234567890abcdef0",
    "Description": "",
    "Enabled": true,
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ],
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeySpec": "SYMMETRIC_DEFAULT",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "MultiRegion": false,
    "Origin": "AWS_CLOUDHSM"
  }
}

```



```
}  
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「[AWS CloudHSM キーストア](#)」を参照してください。

例 8: 外部キーストアに KMS キーを作成するには

次の create-key の例は、指定された外部キーストアに KMS キーを作成します。このコマンドでは、--custom-key-store-id、--origin、および --xks-key-id パラメータが必須です。

--xks-key-id パラメータは、外部キーマネージャーにある既存の対称暗号化キーの ID を指定します。このキーは KMS キーの外部キーマテリアルとして機能します。--origin パラメータの値は、EXTERNAL_KEY_STORE である必要があります。custom-key-store-id パラメータは、外部キーストアプロキシに接続されている外部キーストアを識別する必要があります。

```
aws kms create-key \  
  --origin EXTERNAL_KEY_STORE \  
  --custom-key-store-id cks-9876543210fedcba9 \  
  --xks-key-id bb8562717f809024
```

出力:

```
{  
  "KeyMetadata": {  
    "Arn": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
    "AWSAccountId": "111122223333",  
    "CreationDate": "2022-12-02T07:48:55-07:00",  
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",  
    "CustomKeyId": "ck-9876543210fedcba9",  
    "Description": "",  
    "Enabled": true,  
    "EncryptionAlgorithms": [  
      "SYMMETRIC_DEFAULT"  
    ],  
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",  
    "KeyManager": "CUSTOMER",  
    "KeySpec": "SYMMETRIC_DEFAULT",  
  },  
}
```

```

        "KeyState": "Enabled",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "MultiRegion": false,
        "Origin": "EXTERNAL_KEY_STORE",
        "XksKeyConfiguration": {
            "Id": "bb8562717f809024"
        }
    }
}

```

詳細については、「AWS Key Management Service デベロッパーガイド」の「[外部キーストア](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[CreateKey](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

/**
 * Creates a new symmetric encryption key asynchronously.
 *
 * @param keyDesc the description of the key to be created
 * @return a {@link CompletableFuture} that completes with the ID of the
 * newly created key
 * @throws RuntimeException if an error occurs while creating the key
 */
public CompletableFuture<String> createKeyAsync(String keyDesc) {
    CreateKeyRequest keyRequest = CreateKeyRequest.builder()
        .description(keyDesc)
        .keySpec(KeySpec.SYMMETRIC_DEFAULT)
        .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
        .build();
}

```

```
return getAsyncClient().createKey(keyRequest)
    .thenApply(resp -> resp.keyMetadata().keyId())
    .exceptionally(ex -> {
        throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
    });
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[CreateKey](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun createKey(keyDesc: String?): String? {
    val request =
        CreateKeyRequest {
            description = keyDesc
            customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
            keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[CreateKey](#)」を参照してください。

PHP

SDK for PHP

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keySpec
 * @param string $keyUsage
 * @param string $description
 * @return array
 */
public function createKey(string $keySpec = "", string $keyUsage = "", string
$description = "Created by the SDK for PHP")
{
    $parameters = ['Description' => $description];
    if($keySpec && $keyUsage){
        $parameters['KeySpec'] = $keySpec;
        $parameters['KeyUsage'] = $keyUsage;
    }
    try {
        $result = $this->client->createKey($parameters);
        return $result['KeyMetadata'];
    }catch(KmsException $caught){
        // Check for error specific to createKey operations
        if ($caught->getAwsErrorMessage() == "LimitExceededException"){
            echo "The request was rejected because a quota was exceeded. For
more information, see Quotas in the Key Management Service Developer Guide.";
        }
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[CreateKey](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def create_key(self, key_description: str) -> dict[str, any]:
        """
        Creates a key with a user-provided description.

        :param key_description: A description for the key.
        :return: The key ID.
        """
        try:
            key = self.kms_client.create_key(Description=key_description)
            ["KeyMetadata"]
            self.created_keys.append(key)
            return key
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
```

```
        err.response["Error"]["Message"],
    )
    raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[CreateKey](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
  tags: [
    {
      tag_key: 'CreatedBy',
      tag_value: 'ExampleUser'
    }
  ]
})

puts resp.key_metadata.key_id
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[CreateKey](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn make_key(client: &Client) -> Result<(), Error> {  
    let resp = client.create_key().send().await?;  
  
    let id = resp.key_metadata.as_ref().unwrap().key_id();  
  
    println!("Key: {}", id);  
  
    Ok(())  
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[CreateKey](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **Decrypt**で 使用する

次のサンプルコードは、Decrypt を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

CLI

AWS CLI

例 1: 対称 KMS キーを使用して暗号化されたメッセージを復号するには (Linux および macOS)

次のdecryptコマンド例は、AWS CLI でデータを復号化するための推奨方法を示しています。このバージョンは、対称 KMS キーを使用してデータを復号する方法を示しています。

ファイルに暗号文を指定します。--ciphertext-blob パラメータの値には、バイナリファイルからデータを読み取るように CLI に指示する fileb:// プレフィックスを使用します。ファイルが現在のディレクトリにない場合は、ファイルへのフルパスを入力します。ファイルから CLI AWS パラメータ値を読み取る方法の詳細については、「」を参照してください。AWS 「コマンドラインインターフェイスユーザーガイド」の「ファイル <<https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-parameters-file.html>>」からの AWS CLI パラメータのロード」およびAWS 「コマンドラインツールブログ」の「ローカルファイルパラメータのベストプラクティス<<https://aws.amazon.com/blogs/developer/best-practices-for-local-file-parameters/>>」を参照してください。暗号化テキストを復号する KMS キーを指定します。対称 KMS キーで復号する場合、--key-id パラメータは必要ありません。AWS KMS は、暗号化テキストのメタデータからデータを暗号化するために使用された KMS キーのキー ID を取得できます。ただし、ベストプラクティスは常に、使用している KMS キーを指定することです。この方法により、意図した KMS キーを使用することができ、信頼できない KMS キーを使用して誤って暗号文を復号するのを防ぐことができます。プレーンテキストの出力をテキスト値としてリクエストします。--query パラメータは、出力から Plaintext フィールドの値のみを取得するように CLI に指示します。--output パラメータは出力を text.Base64 でデコードしたプレーンテキストとして返し、ファイルに保存します。次の例では、Plaintext パラメータの値を Base64 ユーティリティにパイプ (|) して、Base64 ユーティリティでデコードします。次に、デコードされた出力を ExamplePlaintext ファイルにリダイレクト (>) します。

このコマンドを実行する前に、サンプルキー ID を AWS アカウントの有効なキー ID に置き換えます。

```
aws kms decrypt \  
  --ciphertext-blob fileb://ExampleEncryptedFile \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --output text \  
  --query Plaintext | base64 \  
> ExamplePlaintext
```



```
--decode > ExamplePlaintextFile
```

このコマンドでは何も出力されません。decrypt コマンドからの出力は base64 でデコードされ、ファイルに保存されます。

詳細については、「AWS Key Management Service API リファレンス」の「[Decrypt](#)」を参照してください。

例 2: 対称 KMS キーを使用して暗号化されたメッセージを復号するには (Windows コマンドプロンプト)

次の例は、certutil ユーティリティを使用してプレーンテキストデータを Base64 でデコードする点を除いて、前の例と同じです。この手順には、次の例に示すように 2 つのコマンドが必要です。

このコマンドを実行する前に、サンプルキー ID を AWS アカウントの有効なキー ID に置き換えます。

```
aws kms decrypt ^
--ciphertext-blob fileb://ExampleEncryptedFile ^
--key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^
--output text ^
--query Plaintext > ExamplePlaintextFile.base64
```

certutil コマンドを実行します。

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

出力:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

詳細については、「AWS Key Management Service API リファレンス」の「[Decrypt](#)」を参照してください。

例 3: 非対称 KMS キーを使用して暗号化されたメッセージを復号するには (Linux および macOS)

次の decrypt コマンドの例は、RSA 非対称 KMS キーで暗号化されたデータを復号する方法を示します。

非対称 KMS キーを使用する場合は、プレーンテキストの暗号化に使用するアルゴリズムを指定する encryption-algorithm パラメータが必要です。

このコマンドを実行する前に、サンプルキー ID を AWS アカウントの有効なキー ID に置き換えます。

```
aws kms decrypt \
  --ciphertext-blob fileb://ExampleEncryptedFile \
  --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
  --encryption-algorithm RSAES_OAEP_SHA_256 \
  --output text \
  --query Plaintext | base64 \
  --decode > ExamplePlaintextFile
```

このコマンドでは何も出力されません。decrypt コマンドからの出力は base64 でデコードされ、ファイルに保存されます。

詳細については、「Key Management Service デベロッパーガイド」の「[KMS AWS の非対称キー](#)」を参照してください。AWS

- API の詳細については、AWS CLI コマンドリファレンスの「[Decrypt](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Asynchronously decrypts the given encrypted data using the specified key
 * ID.
 */
```

```

    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the
    decrypted data as a String.
    *         If an error occurs during the decryption process, the
    CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
    String.
    */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData,
String keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while
decrypting data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Decrypt](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest =
        EncryptRequest {
            keyId = keyIdValue
            plaintext = myBytes
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext
    }
}
```

```
        // Print the decrypted data.
        print(myVal)
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[Decrypt](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param string $ciphertext
 * @param string $algorithm
 * @return Result
 */
public function decrypt(string $keyId, string $ciphertext, string $algorithm
= "SYMMETRIC_DEFAULT")
{
    try{
        return $this->client->decrypt([
            'CiphertextBlob' => $ciphertext,
            'EncryptionAlgorithm' => $algorithm,
            'KeyId' => $keyId,
        ]);
    }catch(KmsException $caught){
        echo "There was a problem decrypting the data: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

```
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[Decrypt](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def decrypt(self, key_id: str, cipher_text: str) -> bytes:
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        :return: The decrypted text.
        """
```

```
        try:
            return self.kms_client.decrypt(KeyId=key_id,
            CiphertextBlob=cipher_text)[
                "Plaintext"
            ]
        except ClientError as err:
            logger.error(
                "Couldn't decrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
```

- APIの詳細については、AWS SDK for Python (Boto3) API リファレンスの「[Decrypt](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Decrypted blob

blob =
  '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593'
blob_packed = [blob].pack('H*')

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.decrypt({
  ciphertext_blob: blob_packed
})
```

```
puts 'Raw text: '  
puts resp.plaintext
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[Decrypt](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),  
Error> {  
    // Open input text file and get contents as a string  
    // input is a base-64 encoded string, so decode it:  
    let data = fs::read_to_string(filename)  
        .map(|input| {  
            base64::decode(input).expect("Input file does not contain valid base  
64 characters.")  
        })  
        .map(Blob::new);  
  
    let resp = client  
        .decrypt()  
        .key_id(key)  
        .ciphertext_blob(data.unwrap())  
        .send()  
        .await?;  
  
    let inner = resp.plaintext.unwrap();  
    let bytes = inner.as_ref();  
  
    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to  
UTF-8");
```



```
println!();  
println!("Decoded string:");  
println!("{}", s);  
  
Ok(())  
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[Decrypt](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteAlias**で を使用する

次のサンプルコードは、DeleteAlias を使用する方法を説明しています。

CLI

AWS CLI

AWS KMS エイリアスを削除するには

次の delete-alias の例は、エイリアス alias/example-alias を削除します。エイリアス名は alias/ で始まる必要があります。

```
aws kms delete-alias \  
  --alias-name alias/example-alias
```

このコマンドは何も出力しません。エイリアスを見つけるには、list-aliases コマンドを使用します。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[エイリアスの削除](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[DeleteAlias](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes a specific KMS alias asynchronously.
 *
 * @param aliasName the name of the alias to be deleted
 * @return a {@link CompletableFuture} representing the asynchronous
 * operation of deleting the specified alias
 */
public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
    DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
        .aliasName(aliasName)
        .build();

    return getAsyncClient().deleteAlias(deleteAliasRequest)
        .thenRun(() -> {
            logger.info("Alias {} has been deleted successfully", aliasName);
        })
        .exceptionally(throwable -> {
            throw new RuntimeException("Failed to delete alias: " +
aliasName, throwable);
        });
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DeleteAlias](#)」を参照してください。

PHP

SDK for PHP

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $aliasName
 * @return void
 */
public function deleteAlias(string $aliasName)
{
    try {
        $this->client->deleteAlias([
            'AliasName' => $aliasName,
        ]);
    } catch (KmsException $caught){
        echo "There was a problem deleting the alias: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[DeleteAlias](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def delete_alias(self, alias: str) -> None:
        """
        Deletes an alias.

        :param alias: The alias to delete.
        """
        try:
            self.kms_client.delete_alias(AliasName=alias)
        except ClientError as err:
            logger.error(
                "Couldn't delete alias %s. Here's why: %s",
                alias,
                err.response["Error"]["Message"],
            )
            raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[DeleteAlias](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeKey**で を使用する

次のサンプルコードは、DescribeKey を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Retrieve information about an AWS Key Management Service (AWS KMS) key.
/// You can supply either the key Id or the key Amazon Resource Name (ARN)
/// to the DescribeKeyRequest KeyId property.
/// </summary>
```

```
public class DescribeKey
{
    public static async Task Main()
    {
        var keyId = "7c9eccc2-38cb-4c4f-9db3-766ee8dd3ad4";
        var request = new DescribeKeyRequest
        {
            KeyId = keyId,
        };

        var client = new AmazonKeyManagementServiceClient();

        var response = await client.DescribeKeyAsync(request);
        var metadata = response.KeyMetadata;

        Console.WriteLine($"{metadata.KeyId} created on:
{metadata.CreationDate}");
        Console.WriteLine($"State: {metadata.KeyState}");
        Console.WriteLine($"{metadata.Description}");
    }
}
```

- 詳細については、AWS SDK for .NET API リファレンスの [DescribeKey](#) を参照してください。

CLI

AWS CLI

例 1: KMS キーの詳細情報を取得するには

次の `describe-key` 例では、サンプルアカウントとリージョンの Amazon S3 の AWS マネージドキーに関する詳細情報を取得します。このコマンドを使用して、AWS マネージドキーとカスターマネージドキーの詳細を検索できます。

KMS キーを指定するには、`key-id` パラメータを使用します。この例では、エイリアス名値を使用していますが、このコマンドでは、キー ID、キー ARN、エイリアス名、またはエイリアス ARN を使用できます。

```
aws kms describe-key \
```

```
--key-id alias/aws/s3
```

出力:

```
{
  "KeyMetadata": {
    "AWSAccountId": "846764612917",
    "KeyId": "b8a9477d-836c-491f-857e-07937918959b",
    "Arn": "arn:aws:kms:us-west-2:846764612917:key/b8a9477d-836c-491f-857e-07937918959b",
    "CurrentKeyMaterialId":
    "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
    "CreationDate": 2017-06-30T21:44:32.140000+00:00,
    "Enabled": true,
    "Description": "Default KMS key that protects my S3 objects when no other
key is defined",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "KeyState": "Enabled",
    "Origin": "AWS_KMS",
    "KeyManager": "AWS",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
      "SYMMETRIC_DEFAULT"
    ]
  }
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「[キーの表示](#)」を参照してください。

例 2: RSA 非対称 KMS キーの詳細を取得するには

次の describe-key の例は、署名と検証に使用される非対称 RSA KMS キーに関する詳細情報を取得します。

```
aws kms describe-key \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

出力:

```
{
  "KeyMetadata": {
```

```

    "AWSAccountId": "111122223333",
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Arn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "CreationDate": "2019-12-02T19:47:14.861000+00:00",
    "CustomerMasterKeySpec": "RSA_2048",
    "Enabled": false,
    "Description": "",
    "KeyState": "Disabled",
    "Origin": "AWS_KMS",
    "MultiRegion": false,
    "KeyManager": "CUSTOMER",
    "KeySpec": "RSA_2048",
    "KeyUsage": "SIGN_VERIFY",
    "SigningAlgorithms": [
        "RSASSA_PKCS1_V1_5_SHA_256",
        "RSASSA_PKCS1_V1_5_SHA_384",
        "RSASSA_PKCS1_V1_5_SHA_512",
        "RSASSA_PSS_SHA_256",
        "RSASSA_PSS_SHA_384",
        "RSASSA_PSS_SHA_512"
    ]
}

```

例 3: マルチリージョンのレプリカキーに関する詳細を取得するには

次の `describe-key` の例は、マルチリージョンのレプリカキーのメタデータを取得します。このマルチリージョンキーは対称暗号化キーです。マルチリージョンキーの `describe-key` コマンドの出力は、プライマリキーとそのすべてのレプリカに関する情報を返します。

```

aws kms describe-key \
  --key-id arn:aws:kms:ap-northeast-1:111122223333:key/  

mrk-1234abcd12ab34cd56ef1234567890ab

```

出力:

```

{
  "KeyMetadata": {
    "MultiRegion": true,
    "AWSAccountId": "111122223333",
    "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/1234abcd12ab34cd56ef1234567890ab",

```



```

    "CreationDate": "2021-06-28T21:09:16.114000+00:00",
    "CurrentKeyMaterialId":
    "0b7fd7ddbac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",
    "Description": "",
    "Enabled": true,
    "KeyId": "mrk-1234abcd12ab34cd56ef1234567890ab",
    "KeyManager": "CUSTOMER",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT_DECRYPT",
    "Origin": "AWS_KMS",
    "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
    "EncryptionAlgorithms": [
        "SYMMETRIC_DEFAULT"
    ],
    "MultiRegionConfiguration": {
        "MultiRegionKeyType": "PRIMARY",
        "PrimaryKey": {
            "Arn": "arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
            "Region": "us-west-2"
        },
        "ReplicaKeys": [
            {
                "Arn": "arn:aws:kms:eu-west-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "eu-west-1"
            },
            {
                "Arn": "arn:aws:kms:ap-northeast-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "ap-northeast-1"
            },
            {
                "Arn": "arn:aws:kms:sa-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab",
                "Region": "sa-east-1"
            }
        ]
    }
}

```

例 4: HMAC KMS キーの詳細を取得するには

次の describe-key の例は、HMAC KMS キーの詳細情報を取得します。

```
aws kms describe-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

出力:

```
{  
  "KeyMetadata": {  
    "AWSAccountId": "123456789012",  
    "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",  
    "Arn": "arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
    "CreationDate": "2022-04-03T22:23:10.194000+00:00",  
    "Enabled": true,  
    "Description": "Test key",  
    "KeyUsage": "GENERATE_VERIFY_MAC",  
    "KeyState": "Enabled",  
    "Origin": "AWS_KMS",  
    "KeyManager": "CUSTOMER",  
    "CustomerMasterKeySpec": "HMAC_256",  
    "MacAlgorithms": [  
      "HMAC_SHA_256"  
    ],  
    "MultiRegion": false  
  }  
}
```

- API の詳細については、AWS CLI コマンドリファレンスの「[DescribeKey](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Asynchronously checks if a specified key is enabled.
 *
 * @param keyId the ID of the key to check
 * @return a {@link CompletableFuture} that, when completed, indicates
 * whether the key is enabled or not
 *
 * @throws RuntimeException if an exception occurs while checking the key
 * state
 */
public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
    DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
        .keyId(keyId)
        .build();

    CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
    return responseFuture.whenComplete((resp, ex) -> {
        if (resp != null) {
            KeyState keyState = resp.keyMetadata().keyState();
            if (keyState == KeyState.ENABLED) {
                logger.info("The key is enabled.");
            } else {
                logger.info("The key is not enabled. Key state: {}",
keyState);
            }
        } else {
            throw new RuntimeException(ex);
        }
    }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DescribeKey](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun describeSpecifcKey(keyIdVal: String?) {
    val request =
        DescribeKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[DescribeKey](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
```

```
* @return array
*/
public function describeKey(string $keyId)
{
    try {
        $result = $this->client->describeKey([
            "KeyId" => $keyId,
        ]);
        return $result['KeyMetadata'];
    } catch (KmsException $caught) {
        if ($caught->getAwsErrorMessage() == "NotFoundException") {
            echo "The request was rejected because the specified entity or
resource could not be found.\n";
        }
        throw $caught;
    }
}
```

- 詳細については、AWS SDK for PHP API リファレンスの [DescribeKey](#) を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
```

```
Creates a KeyManager instance with a default KMS client.

:return: An instance of KeyManager initialized with the default KMS
client.
"""
kms_client = boto3.client("kms")
return cls(kms_client)

def describe_key(self, key_id: str) -> dict[str, any]:
    """
    Describes a key.

    :param key_id: The ARN or ID of the key to describe.
    :return: Information about the key.
    """

    try:
        key = self.kms_client.describe_key(KeyId=key_id)["KeyMetadata"]
        return key
    except ClientError as err:
        logging.error(
            "Couldn't get key '%s'. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )
        raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[DescribeKey](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DisableKey**で を使用する

次のサンプルコードは、DisableKey を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Disable an AWS Key Management Service (AWS KMS) key and then retrieve
/// the key's status to show that it has been disabled.
/// </summary>
public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new DisableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.DisableKeyAsync(request);
```

```
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been disabled.
            var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
            {
                KeyId = keyId,
            });
            Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
        }
    }
}
```

- 詳細については、AWS SDK for .NET API リファレンスの [DisableKey](#) を参照してください。

CLI

AWS CLI

KMS キーを一時的に無効にするには

次の `disable-key` コマンドは、カスタマーマネージド KMS キーを無効にします。KMS キーを再度有効にするには、`enable-key` コマンドを使用します。

```
aws kms disable-key \
    --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

このコマンドは何も出力しません。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[キーの有効化と無効化](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[DisableKey](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Asynchronously disables the specified AWS Key Management Service (KMS)
 * key.
 *
 * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
 * disabled
 * @return a CompletableFuture that, when completed, indicates that the key
 * has been disabled successfully
 */
public CompletableFuture<Void> disableKeyAsync(String keyId) {
    DisableKeyRequest keyRequest = DisableKeyRequest.builder()
        .keyId(keyId)
        .build();

    return getAsyncClient().disableKey(keyRequest)
        .thenRun(() -> {
            logger.info("Key {} has been disabled successfully",keyId);
        })
        .exceptionally(throwable -> {
            throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
        });
}
```

- API 詳細については、「AWS SDK for Java 2.x API リファレンス」の「[DisableKey](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun disableKey(keyIdVal: String?) {
    val request =
        DisableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[DisableKey](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @return void
```

```

    */
    public function disableKey(string $keyId)
    {
        try {
            $this->client->disableKey([
                'KeyId' => $keyId,
            ]);
        } catch (KmsException $caught){
            echo "There was a problem disabling the key: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
}

```

- 詳細については、AWS SDK for PHP API リファレンスの [DisableKey](#) を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")

```

```
return cls(kms_client)

def disable_key(self, key_id: str) -> None:
    try:
        self.kms_client.disable_key(KeyId=key_id)
    except ClientError as err:
        logging.error(
            "Couldn't disable key '%s'. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )
        raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[DisableKey](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **EnableKey**で を使用する

次のサンプルコードは、EnableKey を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// Enable an AWS Key Management Service (AWS KMS) key.
/// </summary>
public class EnableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to enable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new EnableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.EnableKeyAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been enabled.
            var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
            {
                KeyId = keyId,
            });
            Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
        }
    }
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[EnableKey](#)」を参照してください。

CLI

AWS CLI

KMS キーを有効にするには

次の `enable-key` の例は、カスタマーマネージドキーを有効にします。このようなコマンドでは、`disable-key` コマンドを使用して一時的に無効にした KMS キーを有効にできます。また、削除が予定されていて削除がキャンセルされたために無効になっている KMS キーを有効にすることもできます。

KMS キーを指定するには、`key-id` パラメータを使用します。この例では、キー ID 値を使用していますが、このコマンドでは、キー ID またはキー ARN 値を使用できます。

このコマンドを実行する前に、サンプルキー ID を有効なキー ID に置き換えます。

```
aws kms enable-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

このコマンドは何も出力しません。KMS キーが有効になっていることを確認するには、`describe-key` コマンドを使用します。`describe-key` 出力の `KeyState` および `Enabled` フィールドの値を参照してください。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[キーの有効化と無効化](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[EnableKey](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**  
 * Asynchronously enables the specified key.
```

```

    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
    enabled
    */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
            getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred
while enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[EnableKey](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun enableKey(keyIdVal: String?) {
    val request =
        EnableKeyRequest {
            keyId = keyIdVal
        }

    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
        kmsClient.enableKey(request)
        println("$keyIdVal was successfully enabled.")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[EnableKey](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @return void
 */
public function enableKey(string $keyId)
{
    try {
        $this->client->enableKey([
            'KeyId' => $keyId,
        ]);
    } catch (KmsException $caught){
        if($caught->getAwsErrorMessage() == "NotFoundException"){
            echo "The request was rejected because the specified entity or
resource could not be found.\n";
        }
    }
}
```



```
        }  
        throw $caught;  
    }  
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[EnableKey](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    @classmethod  
    def from_client(cls) -> "KeyManager":  
        """  
        Creates a KeyManager instance with a default KMS client.  
  
        :return: An instance of KeyManager initialized with the default KMS  
client.  
        """  
        kms_client = boto3.client("kms")  
        return cls(kms_client)  
  
    def enable_key(self, key_id: str) -> None:  
        """  
        Enables a key. Gets the key state after each state change.
```

```
:param key_id: The ARN or ID of the key to enable.
"""
try:
    self.kms_client.enable_key(KeyId=key_id)
except ClientError as err:
    logging.error(
        "Couldn't enable key '%s'. Here's why: %s",
        key_id,
        err.response["Error"]["Message"],
    )
    raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[EnableKey](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **EnableKeyRotation**で 使用する

次のサンプルコードは、EnableKeyRotation を使用する方法を説明しています。

CLI

AWS CLI

KMS キーの自動ローテーションを有効化するには

次のenable-key-rotation 例では、カスターマネージド KMS キーの自動ローテーションを、ローテーション期間が 180 日となるよう有効化します。KMS キーは、このコマンドの完了日から 1 年後 (約 365 日後) にローテーションされ、その後も毎年自動ローテーションされます。

--key-id パラメータは KMS キーを識別します。この例ではキーの ARN 値を使用していますが、KMS キーのキー ID または ARN のどちらでも使用可能です。--rotation-period-in-days パラメータで次のローテーション日までの日数を指定します。90 日から 2560 日までの範囲で値を指定してください。値を指定しない場合、デフォルトは 365 日になります。

```
aws kms enable-key-rotation \  
  --key-id arn:aws:kms:us-  
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab \  
  --rotation-period-in-days 180
```

このコマンドは何も出力しません。KMS キーが有効になっていることを確認するには、`get-key-rotation-status` コマンドを使用します。

詳細については、「AWS Key Management Service Developer Guide」の「[Rotating keys](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[EnableKeyRotation](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    @classmethod  
    def from_client(cls) -> "KeyManager":  
        """  
        Creates a KeyManager instance with a default KMS client.  
  
        :return: An instance of KeyManager initialized with the default KMS  
client.  
        """  
        kms_client = boto3.client("kms")  
        return cls(kms_client)
```

```
def enable_key_rotation(self, key_id: str) -> None:
    """
    Enables rotation for a key.

    :param key_id: The ARN or ID of the key to enable rotation for.
    """
    try:
        self.kms_client.enable_key_rotation(KeyId=key_id)
    except ClientError as err:
        logging.error(
            "Couldn't enable rotation for key '%s'. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )
        raise
```

- APIの詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[EnableKeyRotation](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **Encrypt**で 使用する

次のサンプルコードは、Encrypt を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

CLI

AWS CLI

例 1: Linux または MacOS でファイルの内容を暗号化するには

次のencryptコマンドは、AWS CLI でデータを暗号化するために推奨される方法を示しています。

```
aws kms encrypt \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --plaintext fileb://ExamplePlaintextFile \  
  --output text \  
  --query CiphertextBlob | base64 \  
  --decode > ExampleEncryptedFile
```

このコマンドは、以下の操作を行います。

--plaintext パラメータを使用して暗号化するデータを指定します。このパラメータ値は base64 でエンコードされている必要があります。plaintext パラメータの値は base64 でエンコードされている必要があります。または、ファイルからバイナリデータを読み取るように AWS CLI に指示する fileb:// プレフィックスを使用する必要があります。ファイルが現在のディレクトリにない場合は、ファイルへのフルパスを入力します。例えば、fileb:///var/tmp/ExamplePlaintextFile や fileb://C:\Temp\ExamplePlaintextFile などです。ファイルから CLI AWS パラメータ値を読み取る方法の詳細については、「」を参照してください。コマンドラインインターフェイスユーザーガイドの「[ファイルからのパラメータのロード](#)」およびコマンドラインツールブログの「[ローカルファイルパラメータのベストプラクティス](#)」を参照してください。--output および AWS --query パラメータを使用してコマンドの出力を制御します。これらのパラメータは暗号化されたデータを抽出します。AWS 暗号文と呼ばれるコマンドの出力から。出力の制御の詳細については、コマンドAWS ラインインターフェイスユーザーガイドの「[コマンド出力の制御](#)」を参照してください。base64 ユーティリティを使用して、抽出された出力をバイナリデータにデコードします。正常な encrypt コマンドによって返される暗号文は base64 でエンコードされたテキストです。AWS CLI を使用して復号する前に、このテキストをデコードする必要があります。バイナリ暗号文をファイルに保存します。コマンド (> ExampleEncryptedFile) の最後の部分では、復号を容易にするためにバイナリ暗号文をファイルに保存します。AWS CLI を使用してデータを復号するコマンドの例については、復号の例を参照してください。

例 2: CLI AWS を使用して Windows でデータを暗号化する

この例は前の例と同じですが、base64 の代わりに certutil ツールを使用します。この手順には、次の例に示すように 2 つのコマンドが必要です。

```
aws kms encrypt \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --plaintext fileb://ExamplePlaintextFile \  
  --output text \  
  --query CiphertextBlob | certutil -encode - > ExampleEncryptedFile
```

```
--query CiphertextBlob > C:\Temp\ExampleEncryptedFile.base64
```

```
certutil -decode C:\Temp\ExampleEncryptedFile.base64 C:\Temp\ExampleEncryptedFile
```

例 3: 非対称 KMS キーを使用して暗号化する

次の encrypt コマンドは、プレーンテキストを非対称 KMS キーで暗号化する方法を示しています。--encryption-algorithm パラメータは必須です。すべての encrypt CLI コマンドと同様に、plaintext パラメータは base64 でエンコードされているか、fileb:// プレフィックスを使用する必要があります。プレフィックスは、ファイルからバイナリデータを読み取るように CLI AWS に指示します。

```
aws kms encrypt \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --encryption-algorithm RSAES_OAEP_SHA_256 \
  --plaintext fileb://ExamplePlaintextFile \
  --output text \
  --query CiphertextBlob | base64 \
  --decode > ExampleEncryptedFile
```

このコマンドでは何も出力されません。

- API の詳細については、AWS CLI コマンドリファレンスの「[Encrypt](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Encrypts the given text asynchronously using the specified KMS client and
 * key ID.
 *
 * @param keyId the ID of the KMS key to use for encryption
 * @param text the text to encrypt
```

```
* @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
*/
public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String
text) {
    SdkBytes myBytes = SdkBytes.fromUtf8String(text);
    EncryptRequest encryptRequest = EncryptRequest.builder()
        .keyId(keyId)
        .plaintext(myBytes)
        .build();

    CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
    return responseFuture.whenComplete((response, ex) -> {
        if (response != null) {
            String algorithm = response.encryptionAlgorithm().toString();
            logger.info("The string was encrypted with algorithm {}.\"",
algorithm);
        } else {
            throw new RuntimeException(ex);
        }
    }).thenApply(EncryptResponse::ciphertextBlob);
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[Encrypt](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()
```

```
val encryptRequest =
    EncryptRequest {
        keyId = keyIdValue
        plaintext = myBytes
    }

KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
    val response = kmsClient.encrypt(encryptRequest)
    val algorithm: String = response.encryptionAlgorithm.toString()
    println("The encryption algorithm is $algorithm")

    // Return the encrypted data.
    return response.ciphertextBlob
}

suspend fun decryptData(
    encryptedDataVal: ByteArray?,
    keyIdVal: String?,
) {
    val decryptRequest =
        DecryptRequest {
            ciphertextBlob = encryptedDataVal
            keyId = keyIdVal
        }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Print the decrypted data.
        print(myVal)
    }
}
```

- APIの詳細については、AWS SDK for Kotlin API リファレンスの「[Encrypt](#)」を参照してください。

PHP

SDK for PHP

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param string $text
 * @return Result
 */
public function encrypt(string $keyId, string $text)
{
    try {
        return $this->client->encrypt([
            'KeyId' => $keyId,
            'Plaintext' => $text,
        ]);
    } catch (KmsException $caught) {
        if ($caught->getAwsErrorMessage() == "DisabledException") {
            echo "The request was rejected because the specified KMS key is
not enabled.\n";
        }
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[Encrypt](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def encrypt(self, key_id: str, text: str) -> str:
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :param text: The text to encrypt.
        :return: The encrypted version of the text.
        """
        try:
            response = self.kms_client.encrypt(KeyId=key_id,
Plaintext=text.encode())
            print(
                f"The string was encrypted with algorithm
{response['EncryptionAlgorithm']}"
            )
            return response["CiphertextBlob"]
```

```
except ClientError as err:
    if err.response["Error"]["Code"] == "DisabledException":
        logger.error(
            "Could not encrypt because the key %s is disabled.", key_id
        )
    else:
        logger.error(
            "Couldn't encrypt text. Here's why: %s",
            err.response["Error"]["Message"],
        )
    raise
```

- APIの詳細については、AWS SDK for Python (Boto3) API リファレンスの「[Encrypt](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

text = '1234567890'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.encrypt({
```

```
        key_id: keyId,
        plaintext: text
    })

# Display a readable version of the resulting encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[Encrypt](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());

    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");
}
```

```
    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }

    Ok(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[Encrypt](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GenerateDataKey**で を使用する

次のサンプルコードは、GenerateDataKey を使用する方法を説明しています。

CLI

AWS CLI

例 1: 256 ビットの対称データキーを生成するには

次のgenerate-data-key例では、 の外部で使用する 256 ビット対称データキーをリクエストします AWS。このコマンドは、すぐに使用および削除できるプレーンテキストのデータキーと、指定された KMS キーで暗号化されたデータキーのコピーを返します。暗号化されたデータとともに、暗号化されたデータキーを安全に保存できます。

256 ビットのデータキーをリクエストするには、値 AES_256 を含む key-spec パラメータを使用します。128 ビットのデータキーをリクエストするには、値 AES_128 を含む key-spec パラメータを使用します。その他のすべてのデータキー長には、number-of-bytes パラメータを使用します。

指定する KMS キーは、対称暗号化 KMS キー、つまりキースペック値が SYMMETRIC_DEFAULT の KMS キーである必要があります。

```
aws kms generate-data-key \
```

```
--key-id alias/ExampleAlias \  
--key-spec AES_256
```

出力:

```
{  
  "Plaintext": "VdzKNHGzUAzJeRBVY+uUmofUGGiDzyB3+i9fVkh3piw=",  
  "KeyId": "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",  
  "KeyMaterialId":  
    "0b7fd7ddb6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6",  
  "CiphertextBlob":  
    "AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk  
+YdhV8MrkBQPeac0ReRVNDt9qlAt+SHgIRF8P0H+7U="
```

Plaintext (プレーンテキストデータキー) と CiphertextBlob (暗号化データキー) は base64 でエンコードされた形式で返されます。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[データキー](#)」を参照してください。例 2: 512 ビットの対称データキーを生成するには

次の generate-data-key の例は、暗号化と復号用の 512 ビットの対称データキーを要求します。このコマンドは、すぐに使用および削除できるプレーンテキストのデータキーと、指定された KMS キーで暗号化されたデータキーのコピーを返します。暗号化されたデータとともに、暗号化されたデータキーを安全に保存できます。

128 ビットまたは 256 ビット以外のキー長をリクエストするには、number-of-bytes パラメーターを使用します。512 ビットのデータキーをリクエストするために、次の例では値が 64 (バイト) の number-of-bytes パラメーターを使用します。

指定する KMS キーは、対称暗号化 KMS キー、つまりキースペック値が SYMMETRIC_DEFAULT の KMS キーである必要があります。

注: この例の出力の値は、表示のために一部が切り捨てられています。

```
aws kms generate-data-key \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --number-of-bytes 64
```

出力:

```
{
  "CiphertextBlob": "AQIBAHi6LtupRpdKl2aJTzkK6Fbh0tQkMlQJJH3PdtHvS/y+hAEnX/
QQNmMwDfg2korNMEc8AAACaDCCAmQGCSqGSIb3DQEHBqCCA1UwggJRAgEAMIICSgYJKoZ...",
  "Plaintext": "ty8Lr0Bk60F07M2Bwt6qbFdNB
+G00ZLtf5MSEb4a13R2UKWG0p06njAwy2n72VRm2m7z/
Pm9Wpbvttz6a4lSo9hgPvKhZ5y6RTm40ovEXiVfBveyX3DQxDzRSwbKDPk/...",
  "KeyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyMaterialId":
  "0b7fd7ddb6ac6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

Plaintext (プレーンテキストデータキー) と CiphertextBlob (暗号化データキー) は base64 でエンコードされた形式で返されます。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[データキー](#)」を参照してください。

- 詳細については、AWS CLI コマンドリファレンスの「[GenerateDataKey](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.
```

```

        :return: An instance of KeyManager initialized with the default KMS
client.
    """
    kms_client = boto3.client("kms")
    return cls(kms_client)

    def generate_data_key(self, key_id):
        """
        Generates a symmetric data key that can be used for client-side
        encryption.
        """
        answer = input(
            f"Do you want to generate a symmetric data key from key {key_id} (y/
n)? "
        )
        if answer.lower() == "y":
            try:
                data_key = self.kms_client.generate_data_key(
                    KeyId=key_id, KeySpec="AES_256"
                )
            except ClientError as err:
                logger.error(
                    "Couldn't generate a data key for key %s. Here's why: %s",
                    key_id,
                    err.response["Error"]["Message"],
                )
            else:
                pprint(data_key)

```

- APIの詳細については、AWS SDK for Python (Boto3) API リファレンスの「[GenerateDataKey](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[GenerateDataKey](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI `GenerateDataKeyWithoutPlaintext`で を使用する

次のサンプルコードは、`GenerateDataKeyWithoutPlaintext` を使用する方法を説明しています。

CLI

AWS CLI

プレーンテキストキーを使用せずに 256 ビットの対称データキーを生成するには

次の `generate-data-key-without-plaintext` の例は、256 ビットの対称データキーの暗号化コピーを AWS の外部で使用するようリクエストします。使用する準備ができたなら、KMS AWS を呼び出してデータキーを復号化できます。

256 ビットのデータキーをリクエストするには、値 `AES_256` を含む `key-spec` パラメータを使用します。128 ビットのデータキーをリクエストするには、値 `AES_128` を含む `key-spec` パラメータを使用します。その他のすべてのデータキー長には、`number-of-bytes` パラメータを使用します。

指定する KMS キーは、対称暗号化 KMS キー、つまりキースペック値が `SYMMETRIC_DEFAULT` の KMS キーである必要があります。

```
aws kms generate-data-key-without-plaintext \
  --key-id "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" \
  --key-spec AES_256
```

出力:

```
{
  "CiphertextBlob":
"AQEDAHjRYf5WytIc0C857tFSnBaPn2F8DgfmThbJlGfR8P3WlwAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgk
  "KeyId": "arn:aws:kms:us-
east-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "KeyMaterialId":
"0b7fd7ddb6eef27907413567cad8c810e2883dc8a7534067a82ee1142fc1e6"
}
```

`CiphertextBlob` (暗号化されたデータキー) は base64 でエンコードされた形式で返されます。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[データキー](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[GenerateDataKeyWithoutPlaintext](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[GenerateDataKeyWithoutPlaintext](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GenerateRandom**で を使用する

次のサンプルコードは、GenerateRandom を使用方法を説明しています。

CLI

AWS CLI

例 1: 256 ビットのランダムバイト文字列を生成するには (Linux または macOS)

次の generate-random の例は、256 ビット (32 バイト) の base64 でエンコードされたランダムバイト文字列を生成します。この例は、バイト文字列をデコードしてランダムファイルに保存します。

このコマンドを実行する際は、number-of-bytes パラメーターを使用してランダム値の長さをバイト単位で指定する必要があります。

このコマンドを実行する際は、KMS キーは指定しません。ランダムバイト文字列は KMS キーとは関係ありません。

デフォルトでは、AWS KMS は乱数を生成します。ただし、[カスタムキーストア](#)を指定すると、カスタムキーストアに関連付けられた AWS CloudHSM クラスターにランダムなバイト文字列が生成されます。

この例は以下のパラメータと値を使用します。

値 の必須--number-of-bytesパラメータを使用して 32 バイト (256 ビット) を32リクエストします。string.It は の値 の--outputパラメータを使用して、出力をテキストとして返すように AWS CLI にtext指示します。代わりに、JSON.It は --query parameter を使用して response.It パイプ (|) から Plaintext プロパティの値を抽出し、コマンドの出力を base64ユーティリティ、抽出された output.It をデコードする は、リダイレクト演算子 (>) を使用してデコードされたバイト文字列を ExampleRandom file.It に保存し、リダイレクト演算子 (>) を使用してバイナリ暗号文をファイルに保存します。

```
aws kms generate-random \  
  --number-of-bytes 32 \  
  --output text \  
  > ExampleRandom file.It
```

```
--query Plaintext | base64 --decode > ExampleRandom
```

このコマンドでは何も出力されません。

詳細については、「AWS Key Management Service API リファレンス」の「[GenerateRandom](#)」を参照してください。

例 2: 256 ビットの乱数を生成するには (Windows コマンドプロンプト)

次の例は、generate-random コマンドを使用して、base64 でエンコードされた 256 ビット (32 バイト) のランダムバイト文字列を生成します。この例は、バイト文字列をデコードしてランダムファイルに保存します。この例は前の例と同じですが、Windows の certutil のユーティリティを使用して、ランダムバイト文字列を base64 でデコードしてからファイルに保存する点が異なります。

最初に base64 でエンコードされたランダムバイト文字列を生成し、テンポラリファイル (ExampleRandom.base64) に保存します。

```
aws kms generate-random \  
  --number-of-bytes 32 \  
  --output text \  
  --query Plaintext > ExampleRandom.base64
```

generate-random コマンドの出力はファイルに保存されるため、この例では何も出力されません。

次に、certutil -decode コマンドを使用して、ExampleRandom.base64 ファイル内の base64 でエンコードされたバイト文字列をデコードします。その後、デコードされたバイト文字列を ExampleRandom ファイルに保存します。

```
certutil -decode ExampleRandom.base64 ExampleRandom
```

出力:

```
Input Length = 18  
Output Length = 12  
CertUtil: -decode command completed successfully.
```

詳細については、「AWS Key Management Service API リファレンス」の「[GenerateRandom](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[GenerateRandom](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[GenerateRandom](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetKeyPolicy**で を使用する

次のサンプルコードは、GetKeyPolicy を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

CLI

AWS CLI

ある KMS キーから別の KMS キーにキーポリシーをコピーするには

次の get-key-policy の例は、ある KMS キーからキーポリシーを取得し、テキストファイルに保存します。その後、テキストファイルをポリシー入力として使用して、別の KMS キーのポリシーを置き換えます。

put-key-policy の --policy パラメータには文字列が必要なため、JSON ではなくテキスト文字列として出力を返す --output text オプションを使用する必要があります。

```
aws kms get-key-policy \
  --policy-name default \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --query Policy \
  --output text > policy.txt

aws kms put-key-policy \
  --policy-name default \
  --key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \
  --policy file://policy.txt
```

このコマンドは何も出力しません。

詳細については、「AWS KMS API リファレンス」の「[PutKeyPolicy](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[GetKeyPolicy](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def get_policy(self, key_id: str) -> dict[str, str]:
        """
        Gets the policy of a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The key policy as a dict.
        """
        if key_id != "":
            try:
                response = self.kms_client.get_key_policy(
                    KeyId=key_id,
                )
                policy = json.loads(response["Policy"])
            except ClientError as err:
                logger.error(
                    "Couldn't get policy for key %. Here's why: %s",

```



```
        key_id,  
        err.response["Error"]["Message"],  
    )  
    raise  
else:  
    pprint(policy)  
    return policy  
else:  
    print("Skipping get policy demo.")
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[GetKeyPolicy](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListAliases**で を使用する

次のサンプルコードは、ListAliases を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
```

```
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// List the AWS Key Management Service (AWS KMS) aliases that have been
defined for
/// the keys in the same AWS Region as the default user. If you want to list
/// the aliases in a different Region, pass the Region to the client
/// constructor.
/// </summary>
public class ListAliases
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListAliasesRequest();
        var response = new ListAliasesResponse();

        do
        {
            response = await client.ListAliasesAsync(request);

            response.Aliases.ForEach(alias =>
            {
                Console.WriteLine($"Created: {alias.CreationDate} Last
Update: {alias.LastUpdatedDate} Name: {alias.AliasName}");
            });

            request.Marker = response.NextMarker;
        }
        while (response.Truncated);
    }
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListAliases](#)」を参照してください。

CLI

AWS CLI

例 1: AWS アカウントとリージョン内のすべてのエイリアスを一覧表示するには

次の例では、`list-aliases` コマンドを使用して、AWS アカウントのデフォルトリージョン内のすべてのエイリアスを一覧表示します。出力には、AWS マネージド KMS キーとカスタマーマネージド KMS キーに関連付けられたエイリアスが含まれます。

```
aws kms list-aliases
```

出力:

```
{
  "Aliases": [
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/testKey",
      "AliasName": "alias/testKey",
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/FinanceDept",
      "AliasName": "alias/FinanceDept",
      "TargetKeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
    },
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/dynamodb",
      "AliasName": "alias/aws/dynamodb",
      "TargetKeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    },
    {
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/aws/ebs",
      "AliasName": "alias/aws/ebs",
      "TargetKeyId": "0987ab65-43cd-21ef-09ab-87654321cdef"
    },
    ...
  ]
}
```

例 2: 特定の KMS キーのすべてのエイリアスを一覧表示するには

次の例では、`list-aliases` コマンドとその `key-id` パラメータを使用して、特定の KMS キーに関連するすべてのエイリアスを一覧表示します。

各エイリアスは 1 つの KMS のみに関連付けられますが、1 つの KMS キーには複数のエイリアスを設定することができます。このコマンドは、KMS AWS コンソールに KMS キーごとに 1 つのエイリアスしか一覧表示されないため、非常に便利です。KMS キーのすべてのエイリアスを検索するには、`list-aliases` コマンドを使用する必要があります。

この例では、KMS キーのキー ID を `--key-id` パラメータに使用しますが、キー ID、キー ARN、エイリアス名、またはエイリアス ARN を使用することもできます。

```
aws kms list-aliases --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

出力:

```
{
  "Aliases": [
    {
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/oregon-test-key",
      "AliasName": "alias/oregon-test-key"
    },
    {
      "TargetKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
      "AliasArn": "arn:aws:kms:us-west-2:111122223333:alias/project121-test",
      "AliasName": "alias/project121-test"
    }
  ]
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「[エイリアスの使用](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[ListAliases](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Asynchronously lists all the aliases in the current AWS account.
 *
 * @return a {@link CompletableFuture} that completes when the list of
 * aliases has been processed
 */
public CompletableFuture<Object> listAllAliasesAsync() {
    ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
        .limit(15)
        .build();

    ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
    return paginator.subscribe(response -> {
        response.aliases().forEach(alias ->
            logger.info("The alias name is: " + alias.aliasName())
        );
    })
        .thenApply(v -> null)
        .exceptionally(ex -> {
            if (ex.getCause() instanceof KmsException) {
                KmsException e = (KmsException) ex.getCause();
                throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
            } else {
                throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
            }
        });
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListAliases](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllAliases() {  
    val request =  
        ListAliasesRequest {  
            limit = 15  
        }  
  
    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.listAliases(request)  
        response.aliases?.forEach { alias ->  
            println("The alias name is ${alias.aliasName}")  
        }  
    }  
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[ListAliases](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param int $limit
 * @return ResultPaginator
 */
public function listAliases(string $keyId = "", int $limit = 0)
{
    $args = [];
    if($keyId){
        $args['KeyId'] = $keyId;
    }
    if($limit){
        $args['Limit'] = $limit;
    }
    try{
        return $this->client->getPaginator("ListAliases", $args);
    }catch(KmsException $caught){
        if($caught->getAwsErrorMessage() == "InvalidMarkerException"){
            echo "The request was rejected because the marker that specifies
where pagination should next begin is not valid.\n";
        }
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListAliases](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    @classmethod
    def from_client(cls) -> "AliasManager":
        """
        Creates an AliasManager instance with a default KMS client.

        :return: An instance of AliasManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def list_aliases(self, page_size: int) -> None:
        """
        Lists aliases for the current account.
        :param page_size: The number of aliases to list per page.
        """
        try:
            alias_paginator = self.kms_client.get_paginator("list_aliases")
            for alias_page in alias_paginator.paginate(
                PaginationConfig={"PageSize": page_size}
            ):
                print(f"Here are {page_size} aliases:")
                pprint(alias_page["Aliases"])
                if alias_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} aliases (y/n)?
"
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your aliases!")
        except ClientError as err:
            logging.error(
                "Couldn't list your aliases. Here's why: %s",
                err.response["Error"]["Message"],
            )

```



```
raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[ListAliases](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListGrants**で を使用する

次のサンプルコードは、ListGrants を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// List the AWS Key Management Service (AWS KMS) grants that are associated
with
/// a specific key.
/// </summary>
```

```
public class ListGrants
{
    public static async Task Main()
    {
        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListGrantsRequest
        {
            KeyId = keyId,
        };

        var response = new ListGrantsResponse();

        do
        {
            response = await client.ListGrantsAsync(request);

            response.Grants.ForEach(grant =>
            {
                Console.WriteLine($"{grant.GrantId}");
            });

            request.Marker = response.NextMarker;
        }
        while (response.Truncated);
    }
}
```

- APIの詳細については、「AWS SDK for .NET API リファレンス」の「[ListGrants](#)」を参照してください。

CLI

AWS CLI

KMS AWS キーの許可を表示するには

次の `list-grants` 例では、アカウントの Amazon DynamoDB の指定された AWS マネージド KMS キーに対するすべての許可を表示します。この権限により、DynamoDB はユー

ザーに代わって KMS キーを使用して、DynamoDB テーブルをディスクに書き込む前に暗号化できます。このようなコマンドを使用して、AWS アカウントとリージョンの AWS マネージド KMS キーとカスターマネージド KMS キーの許可を表示できます。

このコマンドは、キー ID を含む `key-id` パラメータを使用して KMS キーを識別します。ユーザーは、キー ID またはキー ARN を使用して KMS キーを識別できます。AWS マネージド KMS キーのキー ID またはキー ARN を取得するには、`list-keys` または `list-aliases` コマンドを使用します。

```
aws kms list-grants \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

これにより、暗号化操作において Amazon DynamoDB に KMS キーを使用する権限が付与され、KMS キーに関する詳細情報を表示する権限 (`DescribeKey`) と権限を廃止する権限 (`RetireGrant`) が付与されていることが出力に表示されます。EncryptionContextSubset の制約により、これらの権限は指定された暗号化コンテキストペアを含むリクエストに限定されます。そのため、この権限のアクセス許可は指定されたアカウントと DynamoDB テーブルでのみ有効です。

```
{  
  "Grants": [  
    {  
      "Constraints": {  
        "EncryptionContextSubset": {  
          "aws:dynamodb:subscriberId": "123456789012",  
          "aws:dynamodb:tableName": "Services"  
        }  
      },  
      "IssuingAccount": "arn:aws:iam::123456789012:root",  
      "Name": "8276b9a6-6cf0-46f1-b2f0-7993a7f8c89a",  
      "Operations": [  
        "Decrypt",  
        "Encrypt",  
        "GenerateDataKey",  
        "ReEncryptFrom",  
        "ReEncryptTo",  
        "RetireGrant",  
        "DescribeKey"  
      ],  
      "GrantId":  
        "1667b97d27cf748cf05b487217dd4179526c949d14fb3903858e25193253fe59",
```

```

        "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "RetiringPrincipal": "dynamodb.us-west-2.amazonaws.com",
        "GranteePrincipal": "dynamodb.us-west-2.amazonaws.com",
        "CreationDate": "2021-05-13T18:32:45.144000+00:00"
    }
]
}

```

詳細については、AWS 「Key Management Service デベロッパーガイド」の [「KMS AWS でのグラント」](#) を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの [「ListGrants」](#) を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

/**
 * Asynchronously displays the grant IDs for the specified key ID.
 *
 * @param keyId the ID of the AWS KMS key for which to list the grants
 * @return a {@link CompletableFuture} that, when completed, will be null
 * if the operation succeeded, or will throw a {@link RuntimeException} if the
 * operation failed
 * @throws RuntimeException if there was an error listing the grants, either
 * due to an {@link KmsException} or an unexpected error
 */
public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
    ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
        .keyId(keyId)
        .limit(15)
        .build();
}

```

```
ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
return paginator.subscribe(response -> {
    response.grants().forEach(grant -> {
        logger.info("The grant Id is: " + grant.grantId());
    });
})
.thenApply(v -> null)
.exceptionally(ex -> {
    Throwable cause = ex.getCause();
    if (cause instanceof KmsException) {
        throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
    } else {
        throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
    }
});
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListGrants](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun displayGrantIds(keyIdVal: String?) {
    val request =
        ListGrantsRequest {
            keyId = keyIdVal
            limit = 15
        }
}
```

```
KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->
    val response = kmsClient.listGrants(request)
    response.grants?.forEach { grant ->
        println("The grant Id is ${grant.grantId}")
    }
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[ListGrants](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @return Result
 */
public function listGrants(string $keyId)
{
    try{
        return $this->client->listGrants([
            'KeyId' => $keyId,
        ]);
    }catch(KmsException $caught){
        if($caught->getAwsErrorMessage() == "NotFoundException"){
            echo "    The request was rejected because the specified entity
or resource could not be found.\n";
        }
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListGrants](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        try:
            paginator = self.kms_client.get_paginator("list_grants")
```

```
grants = []
page_iterator = paginator.paginate(KeyId=key_id)
for page in page_iterator:
    grants.extend(page["Grants"])

print(f"Grants for key {key_id}:")
pprint(grants)
return grants
except ClientError as err:
    logger.error(
        "Couldn't list grants for key %s. Here's why: %s",
        key_id,
        err.response["Error"]["Message"],
    )
    raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[ListGrants](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListKeyPolicies**で を使用する

次のサンプルコードは、ListKeyPolicies を使用する方法を説明しています。

CLI

AWS CLI

KMS キーのキーポリシーの名前を取得するには

次の list-key-policies の例は、サンプルアカウントとリージョンのカスタマーマネージドキーのキーポリシーの名前を取得します。このコマンドを使用して、AWS マネージドキーとカスタマーマネージドキーのキーポリシーの名前を検索できます。

有効なキーポリシー名は default のみであるため、このコマンドは有用ではありません。

KMS キーを指定するには、`key-id` パラメータを使用します。この例では、キー ID 値を使用していますが、このコマンドでは、キー ID またはキー ARN を使用できます。

```
aws kms list-key-policies \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

出力:

```
{  
  "PolicyNames": [  
    "default"  
  ]  
}
```

KMS キーポリシーの詳細については、AWS 「Key Management Service デベロッパーガイド」の「[KMS AWS でのキーポリシーの使用](#)」を参照してください。AWS

- API の詳細については、AWS CLI コマンドリファレンスの「[ListKeyPolicies](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**  
 * Asynchronously retrieves the key policy for the specified key ID and  
 * policy name.  
 *  
 * @param keyId      the ID of the AWS KMS key for which to retrieve the  
 * policy  
 * @param policyName the name of the key policy to retrieve  
 * @return a {@link CompletableFuture} that, when completed, contains the key  
 * policy as a {@link String}  
 */
```

```
public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
    GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
        .keyId(keyId)
        .policyName(policyName)
        .build();

    return getAsyncClient().getKeyPolicy(policyRequest)
        .thenApply(response -> {
            String policy = response.policy();
            logger.info("The response is: " + policy);
            return policy;
        })
        .exceptionally(ex -> {
            throw new RuntimeException("Failed to get key policy", ex);
        });
}
```

- API の詳細については、「[AWS SDK for Java 2.x API リファレンス](#)」の「ListKeyPolicies」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.
```

```
        :return: An instance of KeyPolicy initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                "PolicyNames"
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
                err.response["Error"]["Message"],
            )
            raise
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- API の詳細については、「AWS SDK for Python (Boto3) APIリファレンス」の「[ListKeyPolicies](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListKeys**で を使用する

次のサンプルコードは、ListKeys を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

.NET

SDK for .NET

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

/// <summary>
/// List the AWS Key Managements Service (AWS KMS) keys for the AWS Region
/// of the default user. To list keys in another AWS Region, supply the
Region
/// as a parameter to the client constructor.
/// </summary>
public class ListKeys
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListKeysRequest();
        var response = new ListKeysResponse();

        do
        {
            response = await client.ListKeysAsync(request);

            response.Keys.ForEach(key =>
            {
                Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
            });

            // Set the Marker property when response.Truncated is true
            // in order to get the next keys.
        }
    }
}
```

```
        request.Marker = response.NextMarker;
    }
    while (response.Truncated);
}
}
```

- API の詳細については、「AWS SDK for .NET API リファレンス」の「[ListKeys](#)」を参照してください。

CLI

AWS CLI

アカウントとリージョンの KMS キーを取得するには

次の `list-keys` の例は、アカウントとリージョンの KMS キーを取得します。このコマンドは、AWS マネージドキーとカスターマネージドキーの両方を返します。

```
aws kms list-keys
```

出力:

```
{
  "Keys": [
    {
      "KeyArn": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
    },
    {
      "KeyArn": "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321",
      "KeyId": "0987dcba-09fe-87dc-65ba-ab0987654321"
    },
    {
      "KeyArn": "arn:aws:kms:us-east-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d",
      "KeyId": "1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}
```

```
]
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「[キーの表示](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[ListKeys](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * 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
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
    }
}
```

```
ListKeysRequest listKeysRequest = ListKeysRequest.builder()
    .limit(15)
    .build();

/*
 * The `subscribe` method is required when using paginator methods in the
AWS SDK
 * because paginator methods return an instance of a `ListKeysPublisher`,
which is
 * based on a reactive stream. This allows asynchronous retrieval of
paginated
 * results as they become available. By subscribing to the stream, we can
process
 * each page of results as they are emitted.
 */
ListKeysPublisher keysPublisher =
kmsAsyncClient.listKeysPaginator(listKeysRequest);
CompletableFuture<Void> future = keysPublisher
    .subscribe(r -> r.keys().forEach(key ->
        System.out.println("The key ARN is: " + key.keyArn() + ". The key
Id is: " + key.keyId()))
    .whenComplete((result, exception) -> {
        if (exception != null) {
            System.err.println("Error occurred: " +
exception.getMessage());
        } else {
            System.out.println("Successfully listed all keys.");
        }
    });

try {
    future.join();
} catch (Exception e) {
    System.err.println("Failed to list keys: " + e.getMessage());
}
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[ListKeys](#)」を参照してください。

Kotlin

SDK for Kotlin

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
suspend fun listAllKeys() {  
    val request =  
        ListKeysRequest {  
            limit = 15  
        }  
  
    KmsClient.fromEnvironment { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.listKeys(request)  
        response.keys?.forEach { key ->  
            println("The key ARN is ${key.keyArn}")  
            println("The key Id is ${key.keyId}")  
        }  
    }  
}
```

- API の詳細については、AWS SDK for Kotlin API リファレンスの「[ListKeys](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。


```
/**
 * @return array
 */
public function listKeys()
{
    try {
        $contents = [];
        $paginator = $this->client->getPaginator("ListKeys");
        foreach($paginator as $result){
            foreach ($result['Content'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }catch(KmsException $caught){
        echo "There was a problem listing the keys: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

- API の詳細については、「AWS SDK for PHP API リファレンス」の「[ListKeys](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []
```

```

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            key_paginator = self.kms_client.get_paginator("list_keys")
            for key_page in key_paginator.paginate(PaginationConfig={"PageSize":
10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page["Keys"])
                if key_page["Truncated"]:
                    answer = input(
                        f"Do you want to see the next {page_size} keys (y/n)? "
                    )
                    if answer.lower() != "y":
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s",
                err.response["Error"]["Message"],
            )

```

- APIの詳細については、AWS SDK for Python (Boto3) API リファレンスの「[ListKeys](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[ListKeys](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **PutKeyPolicy**で を使用する

次のサンプルコードは、PutKeyPolicy を使用する方法を説明しています。

CLI

AWS CLI

KMS キーのキーポリシーを変更するには

次の `put-key-policy` の例は、カスタマーマネージドキーのキーポリシーを変更します。

まず、キーポリシーを作成して、ローカルの JSON ファイルに保存します。この例では、`key_policy.json` がこのファイルです。また、キーポリシーを `policy` パラメータの文字列値として指定することもできます。

このキーポリシーの最初のステートメントは、IAM ポリシーを使用して KMS キーへのアクセスを制御するアクセス許可を AWS アカウントに付与します。2 番目のステートメントは、KMS キーで `describe-key` コマンドおよび `list-keys` コマンドを実行する権限を `test-user` ユーザーに付与します。

`key_policy.json` の内容:

```
{
  "Version" : "2012-10-17",
  "Id" : "key-default-1",
  "Statement" : [
    {
      "Sid" : "Enable IAM User Permissions",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:root"
      },
      "Action" : "kms:*",
      "Resource" : "*"
    },
    {
      "Sid" : "Allow Use of Key",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:user/test-user"
      },
      "Action" : [
        "kms:DescribeKey",
        "kms:ListKeys"
      ],
      "Resource" : "*"
    }
  ]
}
```

```
    }
  ]
}
```

この例では KMS キーの識別にキー ID を使用しますが、キー ARN を使用することもできます。キーポリシーを指定するために、このコマンドは `policy` パラメータを使用します。ポリシーがファイル内にあることを示すには、必要な `file://` プレフィックスを使用します。このプレフィックスは、サポートされているすべてのオペレーティングシステムでファイルを識別するために必要です。最後に、コマンドは値が `default` の `policy-name` パラメータを使用します。ポリシー名を指定しない場合、デフォルト値は `default` となります。唯一の有効な値は `default` です。

```
aws kms put-key-policy \
  --policy-name default \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --policy file://key_policy.json
```

このコマンドでは、出力が生成されません。コマンドが有効であることを確認するには、`get-key-policy` コマンドを使用します。次のコマンド例は、同じ KMS キーのキーポリシーを取得します。値が `text` の `output` パラメータは、読みやすいテキスト形式を返します。

```
aws kms get-key-policy \
  --policy-name default \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --output text
```

出力:

```
{
  "Version" : "2012-10-17",
  "Id" : "key-default-1",
  "Statement" : [
    {
      "Sid" : "Enable IAM User Permissions",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:root"
      },
      "Action" : "kms:*",
```

```
        "Resource" : "*"
    },
    {
        "Sid" : "Allow Use of Key",
        "Effect" : "Allow",
        "Principal" : {
            "AWS" : "arn:aws:iam::111122223333:user/test-user"
        },
        "Action" : [ "kms:Describe", "kms:List" ],
        "Resource" : "*"
    }
]
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「[キーポリシーの変更](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[PutKeyPolicy](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param string $policy
 * @return void
 */
public function putKeyPolicy(string $keyId, string $policy)
{
    try {
        $this->client->putKeyPolicy([
            'KeyId' => $keyId,
            'Policy' => $policy,
```

```

    });
    }catch(KmsException $caught){
        echo "There was a problem replacing the key policy: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}

```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[PutKeyPolicy](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyPolicy":
        """
        Creates a KeyPolicy instance with a default KMS client.

        :return: An instance of KeyPolicy initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def set_policy(self, key_id: str, policy: dict[str, any]) -> None:
        """

```

```

    Sets the policy of a key. Setting a policy entirely overwrites the
    existing
    policy, so care is taken to add a statement to the existing list of
    statements
    rather than simply writing a new policy.

:param key_id: The ARN or ID of the key to set the policy to.
:param policy: The existing policy of the key.
:return: None
"""
principal = input(
    "Enter the ARN of an IAM role to set as the principal on the policy:
"
)
if key_id != "" and principal != "":
    # The updated policy replaces the existing policy. Add a new
statement to
    # the list along with the original policy statements.
    policy["Statement"].append(
        {
            "Sid": "Allow access for ExampleRole",
            "Effect": "Allow",
            "Principal": {"AWS": principal},
            "Action": [
                "kms:Encrypt",
                "kms:GenerateDataKey*",
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:ReEncrypt*",
            ],
            "Resource": "*",
        }
    )
    try:
        self.kms_client.put_key_policy(KeyId=key_id,
Policy=json.dumps(policy))
    except ClientError as err:
        logger.error(
            "Couldn't set policy for key %s. Here's why %s",
            key_id,
            err.response["Error"]["Message"],
        )
        raise
    else:

```



```
        print(f"Set policy for key {key_id}.")
    else:
        print("Skipping set policy demo.")
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンス の「[PutKeyPolicy](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ReEncrypt**で 使用する

次のサンプルコードは、ReEncrypt を使用する方法を説明しています。

CLI

AWS CLI

例 1: 異なる対称 KMS キーで暗号化されたメッセージを再暗号化するには (Linux と macOS)

次のre-encryptコマンド例は、CLI AWS でデータを再暗号化するための推奨方法を示しています。

ファイルに暗号文を指定します。--ciphertext-blob パラメータの値には、バイナリファイルからデータを読み取るように CLI に指示する fileb:// プレフィックスを使用します。ファイルが現在のディレクトリにない場合は、ファイルへのフルパスを入力します。ファイルから AWS CLI パラメータ値を読み取る方法の詳細については、AWS 「コマンドラインインターフェイスユーザーガイド[AWS](#)」の「[ファイルからの CLI パラメータのロード](#)」およびAWS 「コマンドラインツールブログ」の「[ローカルファイルパラメータのベストプラクティス](#)」を参照してください。暗号化テキストを復号するソース KMS キーを指定します。対称暗号化 KMS キーで復号する場合、--source-key-id パラメータは必要ありません。AWS KMS は、暗号化テキスト BLOB のメタデータからデータを暗号化するために使用された KMS キーを取得できます。ただし、ベストプラクティスは常に、使用している KMS キーを指定することです。この方法により、意図した KMS キーを使用することができ、信頼できない KMS キーを使用して誤って暗号文を復号するのを防ぐことができます。送信先の KMS キーを指定して、データを再暗号化します。--destination-key-id パラメータは常に必

須です。この例ではキー ARN を使用していますが、有効である限り任意のキー識別子を使用できます。プレーンテキストの出力をテキスト値としてリクエストします。--query パラメータは、出力から Plaintext フィールドの値のみを取得するよう CLI に命令します。--output パラメータは出力を text.Base64 でデコードしたプレーンテキストとして返し、ファイルに保存します。次の例では、Plaintext パラメータの値を Base64 ユーティリティにパイプ (|) して、Base64 ユーティリティでデコードします。次に、デコードされた出力を ExamplePlaintext ファイルにリダイレクト (>) します。

このコマンドを実行する前に、サンプルキー IDs を AWS アカウントの有効なキー識別子に置き換えます。

```
aws kms re-encrypt \  
  --ciphertext-blob fileb://ExampleEncryptedFile \  
  --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 \  
  --query CiphertextBlob \  
  --output text | base64 --decode > ExampleReEncryptedFile
```

このコマンドでは何も出力されません。re-encrypt コマンドからの出力は base64 でデコードされ、ファイルに保存されます。

詳細については、「AWS Key Management Service API リファレンス」の「[ReEncrypt](#)」を参照してください。

例 2: 異なる対称 KMS キーで暗号化されたメッセージを再暗号化するには (Windows コマンドプロンプト)

次の re-encrypt の例は、certutil ユーティリティを使用してプレーンテキストデータを Base64 でデコードする点を除いて、前の例と同じです。この手順には、次の例に示すように 2 つのコマンドが必要です。

このコマンドを実行する前に、サンプルキー ID を AWS アカウントの有効なキー ID に置き換えます。

```
aws kms re-encrypt ^  
  --ciphertext-blob fileb://ExampleEncryptedFile ^  
  --source-key-id 1234abcd-12ab-34cd-56ef-1234567890ab ^  
  --destination-key-id 0987dcba-09fe-87dc-65ba-ab0987654321 ^  
  --query CiphertextBlob ^  
  --output text > ExampleReEncryptedFile.base64
```

その後、certutil ユーティリティーを使用します。

```
certutil -decode ExamplePlaintextFile.base64 ExamplePlaintextFile
```

出力:

```
Input Length = 18
Output Length = 12
CertUtil: -decode command completed successfully.
```

詳細については、「AWS Key Management Service API リファレンス」の「[ReEncrypt](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[ReEncrypt](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```

def re_encrypt(self, source_key_id, cipher_text):
    """
    Takes ciphertext previously encrypted with one key and reencrypt it by
    using
    another key.

    :param source_key_id: The ARN or ID of the original key used to encrypt
    the
                        ciphertext.
    :param cipher_text: The encrypted ciphertext.
    :return: The ciphertext encrypted by the second key.
    """
    destination_key_id = input(
        f"Your ciphertext is currently encrypted with key {source_key_id}. "
        f"Enter another key ID or ARN to reencrypt it: "
    )
    if destination_key_id != "":
        try:
            cipher_text = self.kms_client.re_encrypt(
                SourceKeyId=source_key_id,
                DestinationKeyId=destination_key_id,
                CiphertextBlob=cipher_text,
            )["CiphertextBlob"]
        except ClientError as err:
            logger.error(
                "Couldn't reencrypt your ciphertext. Here's why: %s",
                err.response["Error"]["Message"],
            )
        else:
            print(f"Reencrypted your ciphertext as: {cipher_text}")
            return cipher_text
    else:
        print("Skipping reencryption demo.")

```

- APIの詳細については、AWS SDK for Python (Boto3) API リファレンスの「[ReEncrypt](#)」を参照してください。

Ruby

SDK for Ruby

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
require 'aws-sdk-kms' # v2: require 'aws-sdk'

# Human-readable version of the ciphertext of the data to reencrypt.

blob =
  '01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593'
sourceCiphertextBlob = [blob].pack('H*')

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = 'arn:aws:kms:us-
west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321'

client = Aws::KMS::Client.new(region: 'us-west-2')

resp = client.re_encrypt({
                        ciphertext_blob: sourceCiphertextBlob,
                        destination_key_id: destinationKeyId
                      })

# Display a readable version of the resulting re-encrypted blob.
puts 'Blob:'
puts resp.ciphertext_blob.unpack('H*')
```

- API の詳細については、「AWS SDK for Ruby API リファレンス」の「[ReEncrypt](#)」を参照してください。

Rust

SDK for Rust

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");
}
```

```
    if verbose {
        println!("Wrote the following to {}: ", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }

    Ok(())
}
```

- API の詳細については、AWS SDK for Rust API リファレンスの「[ReEncrypt](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **RetireGrant**で を使用する

次のサンプルコードは、RetireGrant を使用する方法を説明しています。

CLI

AWS CLI

カスタマーマスターキーの権限を廃止するには

次の retire-grant の例は、KMS キーから権限を削除します。

次のコマンド例は、grant-id パラメータと key-id パラメータを指定します。key-id パラメータの値は、KMS キーのキー ARN である必要があります。

```
aws kms retire-grant \
  --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \
  --key-id arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

このコマンドは何も出力しません。権限が廃止されたことを確認するには、list-grants コマンドを使用します。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[グラントの使用停止と取り消し](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[RetireGrant](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def retire_grant(self, grant):
        """
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant["GrantToken"])
        except ClientError as err:
            logger.error(
```



```
        "Couldn't retire grant %s. Here's why: %s",
        grant["GrantId"],
        err.response["Error"]["Message"],
    )
else:
    print(f"Grant {grant['GrantId']} retired.")
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[RetireGrant](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **RevokeGrant**で を使用する

次のサンプルコードは、RevokeGrant を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

CLI

AWS CLI

カスタマーマスターキーの権限を取り消すには

次の revoke-grant の例は、KMS キーから権限を削除します。次のコマンド例は、grant-id パラメータと key-id パラメータを指定します。key-id パラメータの値には、KMS キーのキー ID またはキー ARN を使用できます。

```
aws kms revoke-grant \  
  --grant-id 1234a2345b8a4e350500d432bccf8ecd6506710e1391880c4f7f7140160c9af3 \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

このコマンドは何も出力しません。権限が取り消されたことを確認するには、list-grants コマンドを使用します。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[グラントの使用停止と取り消し](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[RevokeGrant](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Revokes a grant for the specified AWS KMS key asynchronously.
 *
 * @param keyId    The ID or key ARN of the AWS KMS key.
 * @param grantId  The identifier of the grant to be revoked.
 * @return A {@link CompletableFuture} representing the asynchronous
 * operation of revoking the grant.
 *
 * The {@link CompletableFuture} will complete with a {@link
 * RevokeGrantResponse} object
 *
 * if the operation is successful, or with a {@code null} value if an
 * error occurs.
 */
public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String
keyId, String grantId) {
    RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
        .keyId(keyId)
        .grantId(grantId)
        .build();

    CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
    responseFuture.whenComplete((response, exception) -> {
        if (exception == null) {
            logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
        } else {
```

```
        if (exception instanceof KmsException kmsEx) {
            if (kmsEx.getMessage().contains("Grant does not exist")) {
                logger.info("The grant ID '" + grantId + "' does not
exist. Moving on...");
            } else {
                throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
            }
        } else {
            throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
        }
    }
});

return responseFuture;
}
```

- APIの詳細については、「[AWS SDK for Java 2.x API リファレンス](#)」の「RevokeGrant」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $grantId
 * @param string $keyId
 * @return void
 */
public function revokeGrant(string $grantId, string $keyId)
{
    try{
```

```
$this->client->revokeGrant([
    'GrantId' => $grantId,
    'KeyId' => $keyId,
]);
}catch(KmsException $caught){
    echo "There was a problem with revoking the grant: {$caught-
>getAwsErrorMessage()}.\\n";
    throw $caught;
}
}
```

- API の詳細については、「[AWS SDK for PHP API リファレンス](#)」の「RevokeGrant」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "GrantManager":
        """
        Creates a GrantManager instance with a default KMS client.

        :return: An instance of GrantManager initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```

```
def revoke_grant(self, key_id: str, grant_id: str) -> None:
    """
    Revokes a grant so that it can no longer be used.

    :param key_id: The ARN or ID of the key associated with the grant.
    :param grant_id: The ID of the grant to revoke.
    """
    try:
        self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant_id)
    except ClientError as err:
        logger.error(
            "Couldn't revoke grant %s. Here's why: %s",
            grant_id,
            err.response["Error"]["Message"],
        )
        raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[RevokeGrant](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ScheduleKeyDeletion**で 使用する

次のサンプルコードは、ScheduleKeyDeletion を使用方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

CLI

AWS CLI

カスタマーマネージド KMS キーの削除をスケジュールするには

次の `schedule-key-deletion` の例は、指定されたカスタマーマネージド KMS キーを 15 日後に削除するようにスケジュールします。

`--key-id` パラメータは KMS キーを識別します。この例では、キーの ARN 値を使用していますが、キー ID または KMS キー ARN のいずれかを使用できます。`--pending-window-in-days` パラメータは 7 ~ 30 日の待機期間の長さを指定します。デフォルトでは、待機期間は 30 日です。この例では、値として 15 を指定し、コマンドの完了から 15 日後に AWS に KMS キーを完全に削除するように指示します。

```
aws kms schedule-key-deletion \
  --key-id arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
  --pending-window-in-days 15
```

レスポンスには、キー ARN、キーの状態、待機期間 (`PendingWindowInDays`)、および UNIX 時間の削除日が含まれます。削除日を現地時間に表示するには、KMS AWS コンソールを使用します。PendingDeletion キー状態の KMS キーを暗号化処理で使用することはできません。

```
{
  "KeyId": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "DeletionDate": "2022-06-18T23:43:51.272000+00:00",
  "KeyState": "PendingDeletion",
  "PendingWindowInDays": 15
}
```

詳細については、「AWS Key Management Service デベロッパーガイド」の「[Deleting keys](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[ScheduleKeyDeletion](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Deletes a KMS key asynchronously.
 *
 * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
 * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
 * This means that any files, databases, or other data that were encrypted
using the deleted KMS key
 * will become permanently inaccessible. Exercise extreme caution when
deleting KMS keys.</p>
 *
 * @param keyId the ID of the KMS key to delete
 * @return a {@link CompletableFuture} that completes when the key deletion
is scheduled
 */
public CompletableFuture<Void> deleteKeyAsync(String keyId) {
    ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
        .keyId(keyId)
        .pendingWindowInDays(7)
        .build();

    return getAsyncClient().scheduleKeyDeletion(deletionRequest)
        .thenRun(() -> {
            logger.info("Key {} will be deleted in 7 days", keyId);
        })
        .exceptionally(throwable -> {
            throw new RuntimeException("Failed to schedule key deletion for
key ID: " + keyId, throwable);
        });
}
```

- API の詳細については、「[AWS SDK for Java 2.x API リファレンス](#)」の「ScheduleKeyDeletion」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param int $pendingWindowInDays
 * @return void
 */
public function scheduleKeyDeletion(string $keyId, int $pendingWindowInDays =
7)
{
    try {
        $this->client->scheduleKeyDeletion([
            'KeyId' => $keyId,
            'PendingWindowInDays' => $pendingWindowInDays,
        ]);
    } catch (KmsException $caught){
        echo "There was a problem scheduling the key deletion: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

- API の詳細については、「[AWS SDK for PHP API リファレンス](#)」の「ScheduleKeyDeletion」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":
        """
        Creates a KeyManager instance with a default KMS client.

        :return: An instance of KeyManager initialized with the default KMS
client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)

    def delete_key(self, key_id: str, window: int) -> None:
        """
        Deletes a list of keys.

        Warning:
        Deleting a KMS key is a destructive and potentially dangerous operation.
        When a KMS key is deleted,
        all data that was encrypted under the KMS key is unrecoverable.

        :param key_id: The ARN or ID of the key to delete.
        :param window: The waiting period, in days, before the KMS key is
deleted.
        """

        try:
```

```
        self.kms_client.schedule_key_deletion(
            KeyId=key_id, PendingWindowInDays=window
        )
    except ClientError as err:
        logging.error(
            "Couldn't delete key %s. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )
    raise
```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[ScheduleKeyDeletion](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **Sign**で を使用する

次のサンプルコードは、Sign を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

CLI

AWS CLI

例 1: メッセージ向けデジタル署名を生成するには

次の sign 例では、短いメッセージ向けの暗号化署名を生成します。コマンドの出力には、verify コマンドを使用して検証できる、Base64 エンコードされた Signature フィールドが含まれています。

署名するメッセージと、非対称 KMS キーがサポートする署名アルゴリズムを指定する必要があります。KMS キー向け署名アルゴリズムを取得するには、describe-key コマンドを使用します。

AWS CLI v2 では、message パラメータの値は Base64-encoded されている必要があります。または、メッセージをファイルに保存し、fileb:// プレフィックスを使用して、ファイルからバイナリデータを読み取るように AWS CLI に指示することもできます。

このコマンドを実行する前に、サンプルキー ID を AWS アカウントの有効なキー ID に置き換えます。キー ID は、キー用途が SIGN_VERIFY である非対称 KMS キーを表している必要があります。

```
msg=(echo 'Hello World' | base64)

aws kms sign \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --message fileb://UnsignedMessage \
  --message-type RAW \
  --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256
```

出力:

```
{
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "Signature": "ABCDEFhpyVYyTxbaFE74ccSvEJLJr3zuoV1Hfymz4qv+/fxmxNLA7SE1SiF8lHw80fKZZ3bJ...",
  "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

KMS で非対称 KMS AWS キーを使用する方法の詳細については、「Key Management Service デベロッパーガイド」の「[KMS AWS の非対称キー](#)」を参照してください。AWS

例 2: デジタル署名をファイルに保存するには (Linux および macOS)

次の sign 例では、ローカルファイルに保存されている短いメッセージ向けの暗号化署名を生成します。コマンドはレスポンスから Signature プロパティも取得し、これを Base64 デコードして ExampleSignature ファイルに保存します。この署名ファイルを、署名検証用の verify コマンドで使用できます。

sign コマンドには、Base64 エンコードされたメッセージと、非対称 KMS キーがサポートする署名アルゴリズムが必要です。KMS キーがサポートする署名アルゴリズムを取得するには、describe-key コマンドを使用します。

このコマンドを実行する前に、サンプルキー ID を AWS アカウントの有効なキー ID に置き換えます。キー ID は、キー用途が SIGN_VERIFY である非対称 KMS キーを表している必要があります。

```
echo 'hello world' | base64 > EncodedMessage

aws kms sign \
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \
  --message file:///EncodedMessage \
  --message-type RAW \
  --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \
  --output text \
  --query Signature | base64 --decode > ExampleSignature
```

このコマンドは何も出力しません。この例では、出力の Signature プロパティを抽出し、ファイルに保存します。

KMS で非対称 KMS AWS キーを使用する方法の詳細については、「Key Management Service [AWS デベロッパーガイド](#)」の「[KMS の非対称キー](#)」を参照してください。AWS

- API の詳細については、「AWS CLI コマンドリファレンス」の「[署名](#)」を参照してください。

Java

SDK for Java 2.x

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * Asynchronously signs and verifies data using AWS KMS.
 *
 * <p>The method performs the following steps:
```

```

* <ol>
*   <li>Creates an AWS KMS key with the specified key spec, key usage, and
origin.</li>
*   <li>Signs the provided message using the created KMS key and the
RSASSA-PSS-SHA-256 algorithm.</li>
*   <li>Verifies the signature of the message using the created KMS key
and the RSASSA-PSS-SHA-256 algorithm.</li>
* </ol>
*
* @return a {@link CompletableFuture} that completes with the result of the
signature verification,
*       {@code true} if the signature is valid, {@code false} otherwise.
* @throws KmsException if any error occurs during the KMS operations.
* @throws RuntimeException if an unexpected error occurs.
*/
public CompletableFuture<Boolean> signVerifyDataAsync() {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048)
        .keyUsage(KeyUsageType.SIGN_VERIFY)
        .origin(OriginType.AWS_KMS)
        .build();

    return getAsyncClient().createKey(createKeyRequest)
        .thenCompose(createKeyResponse -> {
            String keyId = createKeyResponse.keyMetadata().keyId();

            SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
            SignRequest signRequest = SignRequest.builder()
                .keyId(keyId)
                .message(messageBytes)
                .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                .build();

            return getAsyncClient().sign(signRequest)
                .thenCompose(signResponse -> {
                    byte[] signedBytes =
signResponse.signature().asByteArray();

                    VerifyRequest verifyRequest = VerifyRequest.builder()
                        .keyId(keyId)

```

```
.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset()))))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

        return getAsyncClient().verify(verifyRequest)
            .thenApply(verifyResponse -> {
                return (boolean) verifyResponse.signatureValid();
            });
    });
}

.throwable);
    });
}
```

- APIの詳細については、「AWS SDK for Java 2.x API リファレンス」の「[署名](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**
 * @param string $keyId
 * @param string $message
 * @param string $algorithm
 * @return Result
 */
```

```

public function sign(string $keyId, string $message, string $algorithm)
{
    try {
        return $this->client->sign([
            'KeyId' => $keyId,
            'Message' => $message,
            'SigningAlgorithm' => $algorithm,
        ]);
    } catch (KmsException $caught){
        echo "There was a problem signing the data: {$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
}

```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[署名](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
        client.

```

```
"""
kms_client = boto3.client("kms")
return cls(kms_client)

def sign(self, key_id: str, message: str) -> str:
    """
    Signs a message with a key.

    :param key_id: The ARN or ID of the key to use for signing.
    :param message: The message to sign.
    :return: The signature of the message.
    """
    try:
        return self.kms_client.sign(
            KeyId=key_id,
            Message=message.encode(),
            SigningAlgorithm="RSASSA_PSS_SHA_256",
        )["Signature"]
    except ClientError as err:
        logger.error(
            "Couldn't sign your message. Here's why: %s",
            err.response["Error"]["Message"],
        )
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[署名](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **TagResource**で を使用する

次のサンプルコードは、TagResource を使用する方法を説明しています。

アクション例は、より大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。次のコード例で、このアクションのコンテキストを確認できます。

- [基本を学ぶ](#)

CLI

AWS CLI

KMS キーにタグを追加するには

次の `tag-resource` 例では、カスターマネージド KMS キーに `"Purpose":"Test"` タグと `"Dept":"IT"` タグを追加します。これらのタグを使用して KMS キーにラベルを付け、アクセス許可と監査用の KMS キーカテゴリを作成できます。

KMS キーを指定するには、`key-id` パラメータを使用します。この例では、キー ID 値を使用していますが、このコマンドでは、キー ID またはキー ARN を使用できます。

```
aws kms tag-resource \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --tags TagKey='Purpose',TagValue='Test' TagKey='Dept',TagValue='IT'
```

このコマンドでは何も出力されません。KMS AWS キーのタグを表示するには、`list-resource-tags` コマンドを使用します。

KMS でのタグの使用の詳細については、「[Key Management Service AWS デベロッパーガイド](#)」の「[キーのタグ付け](#)」を参照してください。AWS

- API の詳細については、AWS CLI コマンドリファレンスの「[TagResource](#)」を参照してください。

Java

SDK for Java 2.x

 Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
/**  
 * Asynchronously tags a KMS key with a specific tag.
```

```
*
* @param keyId the ID of the KMS key to be tagged
* @return a {@link CompletableFuture} that completes when the tagging
operation is finished
*/
public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
    Tag tag = Tag.builder()
        .tagKey("Environment")
        .tagValue("Production")
        .build();

    TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
        .keyId(keyId)
        .tags(tag)
        .build();

    return getAsyncClient().tagResource(tagResourceRequest)
        .thenRun(() -> {
            logger.info("{} key was tagged", keyId);
        })
        .exceptionally(throwable -> {
            throw new RuntimeException("Failed to tag the KMS key",
throwable);
        });
}
```

- API の詳細については、「AWS SDK for Java 2.x API リファレンス」の「[TagResource](#)」を参照してください。

PHP

SDK for PHP

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

    /**
     * @param string $keyId
     * @param array $tags
     * @return void
     */
    public function tagResource(string $keyId, array $tags)
    {
        try {
            $this->client->tagResource([
                'KeyId' => $keyId,
                'Tags' => $tags,
            ]);
        } catch (KmsException $caught) {
            echo "There was a problem applying the tag(s): {$caught->getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
}

```

- APIの詳細については、「AWS SDK for PHP API リファレンス」の「[TagResource](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```

class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    @classmethod
    def from_client(cls) -> "KeyManager":

```

```
"""
Creates a KeyManager instance with a default KMS client.

:return: An instance of KeyManager initialized with the default KMS
client.
"""
kms_client = boto3.client("kms")
return cls(kms_client)

def tag_resource(self, key_id: str, tag_key: str, tag_value: str) -> None:
    """
    Add or edit tags on a customer managed key.

    :param key_id: The ARN or ID of the key to enable rotation for.
    :param tag_key: Key for the tag.
    :param tag_value: Value for the tag.
    """
    try:
        self.kms_client.tag_resource(
            KeyId=key_id, Tags=[{"TagKey": tag_key, "TagValue": tag_value}]
        )
    except ClientError as err:
        logging.error(
            "Couldn't add a tag for the key '%s'. Here's why: %s",
            key_id,
            err.response["Error"]["Message"],
        )
        raise
```

- APIの詳細については、AWS SDK for Python (Boto3) API リファレンスの「[TagResource](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDKでのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **UpdateAlias**で を使用する

次のサンプルコードは、UpdateAlias を使用する方法を説明しています。

CLI

AWS CLI

エイリアスを別の KMS キーに関連付けるには

次の `update-alias` の例は、エイリアス `alias/test-key` を別の KMS キーに関連付けます。

`--alias-name` パラメータはエイリアス名を指定します。エイリアス名の値は `alias/` で始まる必要があります。`--target-key-id` パラメータは、エイリアスに関連付ける KMS キーを指定します。エイリアスの現在の KMS キーを指定する必要はありません。

```
aws kms update-alias \  
  --alias-name alias/test-key \  
  --target-key-id 1234abcd-12ab-34cd-56ef-1234567890ab
```

このコマンドは何も出力しません。エイリアスを見つけるには、`list-aliases` コマンドを使用します。

詳細については、「AWS Key Management Service デベロッパーガイド」の「[エイリアスの更新](#)」を参照してください。

- API の詳細については、AWS CLI コマンドリファレンスの「[UpdateAlias](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class AliasManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_key = None
```

```

@classmethod
def from_client(cls) -> "AliasManager":
    """
    Creates an AliasManager instance with a default KMS client.

    :return: An instance of AliasManager initialized with the default KMS
client.
    """
    kms_client = boto3.client("kms")
    return cls(kms_client)

def update_alias(self, alias, current_key_id):
    """
    Updates an alias by assigning it to another key.

    :param alias: The alias to reassign.
    :param current_key_id: The ARN or ID of the key currently associated with
the alias.
    """
    new_key_id = input(
        f"Alias {alias} is currently associated with {current_key_id}. "
        f"Enter another key ID or ARN that you want to associate with
{alias}: "
    )
    if new_key_id != "":
        try:
            self.kms_client.update_alias(AliasName=alias,
TargetKeyId=new_key_id)
        except ClientError as err:
            logger.error(
                "Couldn't associate alias %s with key %s. Here's why: %s",
                alias,
                new_key_id,
                err.response["Error"]["Message"],
            )
        else:
            print(f"Alias {alias} is now associated with key {new_key_id}.")
    else:
        print("Skipping alias update.")

```

- API の詳細については、AWS SDK for Python (Boto3) API リファレンスの「[UpdateAlias](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **Verify**で を使用する

次のサンプルコードは、Verify を使用する方法を説明しています。

CLI

AWS CLI

デジタル署名を検証するには

次の verify は、短い Base64 でエンコードされたメッセージの暗号化署名を検証します。キー ID、メッセージ、メッセージタイプ、署名アルゴリズムは、メッセージの署名に使用されたものと同じである必要があります。

AWS CLI v2 では、message パラメータの値は Base64-encoded されている必要があります。または、メッセージをファイルに保存し、fileb:// プレフィックスを使用して、ファイルからバイナリデータを読み取るように AWS CLI に指示することもできます。

指定する署名は Base64 エンコードされているものであってはなりません。sign コマンドが返す署名のデコードについては、sign コマンドの例を参照してください。

コマンドの出力には、署名が検証済みであることを示すブール値 SignatureValid フィールドが含まれます。署名の検証が失敗した場合、verify コマンドの実行結果も失敗となります。

このコマンドを実行する前に、サンプルキー ID を AWS アカウントの有効なキー ID に置き換えます。

```
aws kms verify \  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab \  
  --message fileb://EncodedMessage \  
  --message-type RAW \  
  --signing-algorithm RSASSA_PKCS1_V1_5_SHA_256 \  
  --signature fileb://ExampleSignature
```

出力:

```
{
  "KeyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "SignatureValid": true,
  "SigningAlgorithm": "RSASSA_PKCS1_V1_5_SHA_256"
}
```

KMS で非対称 KMS AWS キーを使用する方法の詳細については、AWS 「Key Management Service [デベロッパーガイド](#)」の「[非対称キーの使用](#)」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「[Verify](#)」を参照してください。

Python

SDK for Python (Boto3)

Note

GitHub には、その他のリソースもあります。[AWS コード例リポジトリ](#) で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    @classmethod
    def from_client(cls) -> "KeyEncrypt":
        """
        Creates a KeyEncrypt instance with a default KMS client.

        :return: An instance of KeyEncrypt initialized with the default KMS
        client.
        """
        kms_client = boto3.client("kms")
        return cls(kms_client)
```



```
def verify(self, key_id: str, message: str, signature: str) -> bool:
    """
    Verifies a signature against a message.

    :param key_id: The ARN or ID of the key used to sign the message.
    :param message: The message to verify.
    :param signature: The signature to verify.
    :return: True when the signature matches the message, otherwise False.
    """
    try:
        response = self.kms_client.verify(
            KeyId=key_id,
            Message=message.encode(),
            Signature=signature,
            SigningAlgorithm="RSASSA_PSS_SHA_256",
        )
        valid = response["SignatureValid"]
        print(f"The signature is {'valid' if valid else 'invalid'}.")
        return valid
    except ClientError as err:
        if err.response["Error"]["Code"] == "SignatureDoesNotMatchException":
            print("The signature is not valid.")
        else:
            logger.error(
                "Couldn't verify your signature. Here's why: %s",
                err.response["Error"]["Message"],
            )
        raise
```

- API の詳細については、「AWS SDK for Python (Boto3) API リファレンス」の「[Verify](#)」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs AWS KMS を使用するシナリオ

次のコード例は、AWS SDKs を使用して AWS KMS 一般的なシナリオを実装する方法を示しています。これらのシナリオでは、内で複数の関数を呼び出す AWS KMS か、他の関数と組み合わせて特定のタスクを実行する方法を示します AWS のサービス。各シナリオには、完全なソースコードへのリンクが含まれており、そこからコードの設定方法と実行方法に関する手順を確認できます。

シナリオは、サービスアクションをコンテキストで理解するのに役立つ中級レベルの経験を対象としています。

例

- [v2 を使用した DynamoDB AWS Command Line Interface テーブル暗号化の操作](#)

v2 を使用した DynamoDB AWS Command Line Interface テーブル暗号化の操作

次のコード例は、DynamoDB テーブルの暗号化オプションを管理する方法を示しています。

- デフォルトの暗号化を使用してテーブルを作成します。
- カスタマーマネージド CMK をを使用してテーブルを作成します。
- テーブルの暗号化設定を更新します。
- テーブルの暗号化について説明します。

Bash

AWS CLI Bash スクリプトを使用する

デフォルトの暗号化を使用してテーブルを作成します。

```
# Create a table with default encryption (AWS owned key)
aws dynamodb create-table \
  --table-name CustomerData \
  --attribute-definitions \
    AttributeName=CustomerID,AttributeType=S \
  --key-schema \
    AttributeName=CustomerID,KeyType=HASH \
  --billing-mode PAY_PER_REQUEST \
```

```
--sse-specification Enabled=true,SSEType=KMS
```

カスタマーマネージド CMK を使用してテーブルを作成します。

```
# Step 1: Create a customer managed key in KMS
aws kms create-key \
  --description "Key for DynamoDB table encryption" \
  --key-usage ENCRYPT_DECRYPT \
  --customer-master-key-spec SYMMETRIC_DEFAULT

# Store the key ID for later use
KEY_ID=$(aws kms list-keys --query "Keys[?contains(KeyArn, 'Key for
DynamoDB')].KeyId" --output text)

# Step 2: Create a table with the customer managed key
aws dynamodb create-table \
  --table-name SensitiveData \
  --attribute-definitions \
    AttributeName=RecordID,AttributeType=S \
  --key-schema \
    AttributeName=RecordID,KeyType=HASH \
  --billing-mode PAY_PER_REQUEST \
  --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

テーブルの暗号化を更新します。

```
# Update a table to use a different KMS key
aws dynamodb update-table \
  --table-name CustomerData \
  --sse-specification Enabled=true,SSEType=KMS,KMSMasterKeyId=$KEY_ID
```

テーブルの暗号化について説明します。

```
# Describe the table to see encryption settings
aws dynamodb describe-table \
  --table-name CustomerData \
  --query "Table.SSEDescription"
```

- API の詳細については、「AWS CLI コマンドリファレンス」の以下のトピックを参照してください。
 - [CreateKey](#)
 - [CreateTable](#)
 - [DescribeTable](#)
 - [UpdateTable](#)

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してください[AWS SDK でのこのサービスの使用](#)。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS Nitro Enclaves の暗号化アテステーション

AWS KMS は[AWS](#)、[Nitro Enclaves](#) の暗号化アテステーションをサポートしています。AWS Nitro Enclaves をサポートするアプリケーションは、エンクレーブの署名付きアテステーションドキュメントを使用して次の AWS KMS 暗号化オペレーションを呼び出します。これらの AWS KMS APIs、アテステーションドキュメントが Nitro Enclave からのものであることを検証します。次にこれらの API は、レスポンスでプレーンテキストデータを返す代わりに、アテステーションドキュメントのパブリックキーを使用してプレーンテキストを暗号化し、エンクレーブ内の対応するプライベートキーによってのみ復号できる暗号文を返します。

- [Decrypt](#)
- [DeriveSharedSecret](#)
- [GenerateDataKey](#)
- [GenerateDataKeyPair](#)
- [GenerateRandom](#)

次の表は、Nitro Enclave リクエストへのレスポンスが各 API オペレーションの標準的なレスポンスとどのように異なるかを示しています。

AWS KMS オペレーション	標準的なレスポンス	AWS Nitro Enclaves のレスポンス
Decrypt	プレーンテキストデータを返します	アテステーションドキュメントのパブリックキーによって暗号化されたプレーンテキストデータを返します
DeriveSharedSecret	未加工の共有シークレットを返す	アテステーションドキュメントのパブリックキーによって暗号化された未加工の共有シークレットを返す
GenerateDataKey	データキーのプレーンテキストのコピーを返します	アテステーションドキュメントのパブリックキーによって暗号化されたデータキーのコピーを返します

AWS KMS オペレーション	標準的なレスポンス	AWS Nitro Enclaves のレスポンス
	(KMS キーによって暗号化されたデータキーのコピーも返します)	(KMS キーによって暗号化されたデータキーのコピーも返します)
GenerateDataKeyPair	プライベートキーのプレーンテキストのコピーを返します (パブリックキーおよび KMS キーによって暗号化されたプライベートキーのコピーも返します)	アテステーションドキュメントのパブリックキーによって暗号化されたプライベートキーのコピーを返します (パブリックキーおよび KMS キーによって暗号化されたプライベートキーのコピーも返します)
GenerateRandom	乱数バイト文字列を返します	アテステーションドキュメントのパブリックキーによって暗号化された乱数バイト文字列を返します

AWS KMS は、アテステーションドキュメントの内容に基づいて キーでエンクレーブオペレーションを許可または拒否するために使用できる[ポリシー条件](#) AWS KMS キーをサポートします。AWS CloudTrail ログで [Nitro Enclave AWS KMS の へのリクエストをモニタリング](#)することもできます。

詳細はこちら

- [暗号化アテステーション](#)
- [AWS KMSAWS Nitro Enclaves の条件キー](#)
- [Nitro Enclave AWS KMS APIs を呼び出す方法](#)
- [Nitro Enclaves に対するリクエストの監視](#)

Nitro Enclave AWS KMS APIs を呼び出す方法

Nitro Enclave AWS KMS APIs を呼び出すには、リクエストで Recipientパラメータを使用して、エンクレーブの署名付きアテステーションドキュメントと、エンクレーブのパブリックキーで使用する

る暗号化アルゴリズムを指定します。リクエストに Recipient パラメーターと署名付きアテステーションドキュメントが含まれている場合、レスポンスには CiphertextForRecipient フィールドとパブリックキーによって暗号化された暗号文が含まれます。プレーンテキストフィールドは null または空です。

Recipient パラメータは、AWS Nitro enclave. AWS KMS relies からエンクレーブのアテステーションドキュメントの署名付きアテステーションドキュメントを指定して、リクエストのパブリックキーが有効なエンクレーブから送信されたことを証明する必要があります。アテステーションドキュメントにデジタル署名をして独自の証明書を提供することはできません。

Recipient パラメーターを指定するには、[AWS Nitro Enclaves SDK](#) または任意の AWS SDK を使用します。AWS Nitro Enclave 内でのみサポートされている Nitro Enclaves SDK は、すべての AWS KMS リクエストに Recipient パラメータとその値を自動的に追加します。AWS SDKs で Nitro エンクレーブをリクエストするには、Recipient パラメータとその値を指定する必要があります。AWS SDKs での Nitro Enclave 暗号化認証のサポートは、2023 年 3 月に導入されました。

AWS KMS は、アテステーションドキュメントの内容に基づいて キーでエンクレーブオペレーションを許可または拒否するために使用できる[ポリシー条件](#) AWS KMS キーをサポートします。[Nitro Enclave AWS KMS の へのリクエストをログでモニタリング](#)することもできます。AWS CloudTrail

Recipient パラメータと AWS CiphertextForRecipient レスポンスフィールドの詳細については、AWS Key Management Service API リファレンスの[Decrypt](#)、[DeriveSharedSecret](#)、[GenerateDataKey](#)、[GenerateDataKeyPair](#)、[GenerateRandom](#) トピック、[AWS Nitro Enclaves SDK](#)、または任意の AWS SDK を参照してください。暗号化用のデータとデータキーの設定については、「[での暗号化アテステーションの使用 AWS KMS](#)」を参照してください。

Nitro Enclaves に対するリクエストの監視

AWS CloudTrail ログを使用して、AWS Nitro Enclave の[Decrypt](#)、[DeriveSharedSecret](#)、[GenerateDataKey](#)、[GenerateDataKeyPair](#)、および[GenerateRandom](#) オペレーションをモニタリングできます。これらのログエントリの additionalEventData フィールドには、リクエスト内のアテステーションドキュメントからのモジュール ID (attestationDocumentModuleId)、イメージダイジェスト (attestationDocumentEnclaveImageDigest)、およびプラットフォーム設定登録 (PCR) を含む recipient フィールドがあります。これらのフィールドは、リクエストの Recipient パラメータが AWS Nitro Enclave から署名付きアテステーションドキュメントを指定する場合にのみ含まれます。

モジュール ID は Nitro Enclave の [エンクレーブ ID](#) です。イメージダイジェストは、エンクレーブイメージの SHA384 ハッシュです。 [キーポリシーと IAM ポリシーの条件](#) でイメージダイジェストおよび PCR 値を使用できます。PCR の詳細については、『AWS Nitro Enclaves ユーザーガイド』の「[エンクレーブの測定値の入手方法](#)」を参照してください。

このセクションでは、AWS KMS に対するサポートされている Nitro Enclave リクエストそれぞれの CloudTrail ログエントリの例を示します。

Decrypt (エンクレーブ用)

次の例は、AWS Nitro Enclave の [Decrypt](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2020-07-27T22:58:24Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  "responseElements": null,
  "additionalEventData": {
    "recipient": {
      "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
      "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
      "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
      "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
      "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
      "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>"
    }
  }
}
```



```

      "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
    },
    },
    "requestID": "b4a65126-30d5-4b28-98b9-9153da559963",
    "eventID": "e5a2f202-ba1a-467c-b4ba-f729d45ae521",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
      }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
  }
}

```

GenerateDataKey (エンクレーブ用)

次の例は、AWS Nitro Enclave の [GenerateDataKey](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2014-11-04T00:52:40Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "numberOfBytes": 32
  }
},

```

```

"responseElements": null,
"additionalEventData": {
  "recipient": {
    "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
    "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
    "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
    "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
    "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
    "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
    "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
  }
},
"requestID": "e0eb83e3-63bc-11e4-bc2b-4198b6150d5c",
"eventID": "a9dea4f9-8395-46c0-942c-f509c02c2b71",
"readOnly": true,
"resources": [{
  "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
  "accountId": "111122223333"
}],
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

GenerateDataKeyPair (エンクレーブ用)

次の例は、AWS Nitro Enclave の [GenerateDataKeyPair](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::111122223333:user/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2020-07-27T18:57:57Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKeyPair",
  "awsRegion": "us-west-2",

```

```

"sourceIPAddress": "192.0.2.0",
"userAgent": "AWS Internal",
"requestParameters": {
  "keyPairSpec": "RSA_3072",
  "encryptionContext": {
    "Project": "Alpha"
  },
  "keyId": "1234abcd-12ab-34cd-56ef-1234567890ab"
},
"responseElements": null,
"additionalEventData": {
  "recipient": {
    "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
    "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
    "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
    "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
    "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
    "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
    "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
  }
},
"requestID": "52fb127b-0fe5-42bb-8e5e-f560febde6b0",
"eventID": "9b6bd6d2-529d-4890-a949-593b13800ad7",
"readOnly": true,
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }
],
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

GenerateRandom (エンクレーブ用)

次の例は、AWS Nitro Enclave の [GenerateRandom](#) オペレーションの AWS CloudTrail ログエントリを示しています。

```

{
  "eventVersion": "1.02",

```

```
"userIdentity": {
  "type": "IAMUser",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::111122223333:user/Alice",
  "accountId": "111122223333",
  "accessKeyId": "EXAMPLE_KEY_ID",
  "userName": "Alice"
},
"eventTime": "2014-11-04T00:52:37Z",
"eventSource": "kms.amazonaws.com",
"eventName": "GenerateRandom",
"awsRegion": "us-east-1",
"sourceIPAddress": "192.0.2.0",
"userAgent": "AWS Internal",
"requestParameters": null,
"responseElements": null,
"additionalEventData": {
  "recipient": {
    "attestationDocumentModuleId": "i-123456789abcde123-enc123456789abcde12",
    "attestationDocumentEnclaveImageDigest": "<AttestationDocument.PCR0>",
    "attestationDocumentEnclavePCR1": "<AttestationDocument.PCR1>",
    "attestationDocumentEnclavePCR2": "<AttestationDocument.PCR2>",
    "attestationDocumentEnclavePCR3": "<AttestationDocument.PCR3>",
    "attestationDocumentEnclavePCR4": "<AttestationDocument.PCR4>",
    "attestationDocumentEnclavePCR8": "<AttestationDocument.PCR8>"
  }
},
"requestID": "df1e3de6-63bc-11e4-bc2b-4198b6150d5c",
"eventID": "239cb9f7-ae05-4c94-9221-6ea30eef0442",
"readOnly": true,
"resources": [],
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}
```

AWS サービスでの AWS KMS 暗号化の使用

を使用すると AWS Key Management Service、他の AWS サービスのデータを保護するための暗号化キーを提供できます。AWS サービスとの AWS KMS 暗号化の使用とは、AWS KMS 他の AWS サービスと統合して、保管中または転送中のデータを暗号化および復号するプロセスを指します。開発者、システム管理者、およびセキュリティプロフェッショナルは、AWS サービスを通じて保存または送信される機密データを保護し、規制コンプライアンス要件を満たし、暗号化のベストプラクティスを実装するために、このトピックに関心を持つ場合があります。一般的なユースケースには、Amazon EBS ボリューム、Amazon S3 バケット、および Amazon RDS データベースの暗号化があります。以下のセクションでは、特定の AWS サービスの AWS KMS 暗号化キーを設定および管理し、AWS 環境全体でデータの機密性と整合性を確保する手順について説明します。と統合された AWS サービスの完全なリストについては AWS KMS、[AWS 「サービス統合」](#) を参照してください。

以下のトピックでは AWS KMS、サポートされる KMS キー、データキーの管理方法、必要なアクセス許可、アカウント内の各サービスによる KMS キーの使用を追跡する方法など、特定の のサービスがどのように を使用するかについて詳しく説明します。

Important

[AWS と統合された サービスは、AWS KMS](#) 対称暗号化 KMS キーのみを使用してデータを暗号化します。これらのサービスは、非対称 KMS キーを使用する暗号化をサポートしません。KMS キーが対称か非対称かを判断する方法については、[さまざまなキータイプの特定](#) を参照してください。

トピック

- [Amazon Elastic Block Store \(Amazon EBS\) が を使用する 方法 AWS KMS](#)
- [Amazon EMR が を使用する 方法 AWS KMS](#)
- [Amazon Redshift が を使用する 方法 AWS KMS](#)

Amazon Elastic Block Store (Amazon EBS) が を使用する方法

AWS KMS

このトピックでは、[Amazon Elastic Block Store \(Amazon EBS\)](#) が を使用してボリュームとスナップショット AWS KMS を暗号化する方法について詳しく説明します。Amazon EBS ボリュームの暗号化に関する基本的な手順については、「[Amazon EBS 暗号化](#)」を参照してください。

トピック

- [Amazon EBS 暗号化](#)
- [KMS キーとデータキーを使用する](#)
- [Amazon EBS 暗号化コンテキスト](#)
- [Amazon EBS 障害の検出](#)
- [AWS CloudFormation を使用して暗号化された Amazon EBS ボリュームを作成する](#)

Amazon EBS 暗号化

暗号化された Amazon EBS ボリューム [をサポートされている Amazon Elastic Compute Cloud \(Amazon EC2 \) インスタンスタイプ](#) にアタッチすると、ボリュームに保存されたデータ、ディスク I/O、ボリュームから作成されたスナップショットはすべて暗号化されます。暗号化は、Amazon EC2 インスタンスをホストするサーバーで行われます。

この機能は、すべての [Amazon EBS ボリュームタイプ](#) でサポートされています。暗号化されたボリュームには、他のボリュームにアクセスする場合と同じ方法でアクセスできます。暗号化と復号化は透過的に処理され、ユーザー、EC2 インスタンス、アプリケーションからの追加アクションは不要です。暗号化されたボリュームのスナップショットは自動的に暗号化され、暗号化されたスナップショットから作成されたボリュームも、自動的に暗号化されます。

EBS ボリュームの暗号化ステータスは、ボリュームの作成時に決定されます。既存のボリュームの暗号化ステータスを変更することはできません。ただし、暗号化されたボリュームと暗号化されていないボリューム間で [データを移行](#) し、スナップショットのコピー中に新しい暗号化ステータスを適用できます。

Amazon EBS では、デフォルトでオプションの暗号化がサポートされています。AWS アカウントおよびリージョンのすべての新しい EBS ボリュームとスナップショットコピーで、暗号化を自動的に有効にできます。この構成設定は、既存のボリュームやスナップショットには影響しません。詳細については、「[Amazon EBS ユーザーガイド](#)」の「Amazon EBS 暗号化」を参照してください。

KMS キーとデータキーを使用する

[暗号化 Amazon EBS ボリュームを作成する](#)ときは、AWS KMS keyを指定します。デフォルトでは、Amazon EBS はアカウント (aws/ebs) の Amazon EBS 用 [AWS マネージドキー](#) を使用します。ただし、ユーザーは作成および管理する [カスタマーマネージドキー](#) を指定することができます。

カスタマーマネージドキーを使用するには、ユーザーに代わって KMS キーを使用する許可を Amazon EBS に付与する必要があります。必要なアクセス許可のリストについては、「[Amazon EC2 ユーザーガイド](#)」の「IAM ユーザー向けアクセス許可」、または「[Amazon EC2 ユーザーガイド](#)」を参照してください。

Important

Amazon EBS は、[対称 KMS キー](#)のみをサポートします。[非対称 KMS キー](#)を使用して Amazon EBS ボリュームを暗号化することはできません。KMS キーが対称か非対称かを判断する方法については、[さまざまなキータイプの特定](#) を参照してください。

ボリュームごとに、Amazon EBS は指定した KMS キーで暗号化された一意のデータキーを生成する AWS KMS ように 求めます。Amazon EBS は、暗号化されたデータキーをボリュームとともに保存します。次に、ボリュームを Amazon EC2 インスタンスにアタッチすると、Amazon EBS は AWS KMS を呼び出してデータキーを復号します。Amazon EBS は、ハイパーバイザーメモリ内のプレーンテキストデータキーを使用して、ボリュームへのすべてのディスク I/O を暗号化します。詳細については、[Amazon EC2 ユーザーガイド](#) または「Amazon EC2 ユーザーガイド」の「EBS 暗号化の仕組み」を参照してください。 [Amazon EC2](#)

Amazon EBS 暗号化コンテキスト

[GenerateDataKeyWithoutPlaintext](#) および [Decrypt](#) リクエストでは AWS KMS、Amazon EBS はリクエスト内のボリュームまたはスナップショットを識別する名前と値のペアを持つ暗号化コンテキストを使用します。暗号化コンテキストの名前は変わりません。

[暗号化コンテキスト](#) は、一連のキー値のペアおよび任意非シークレットデータを含みます。データを暗号化するリクエストに暗号化コンテキストを含めると、 は暗号化コンテキストを暗号化されたデータに AWS KMS 暗号化バインドします。データを復号するには、同じ暗号化コンテキストに渡す必要があります。

すべてのボリュームと Amazon EBS [CreateSnapshot](#) オペレーションで作成された暗号化されたスナップショットの場合、Amazon EBS はボリューム ID を暗号化コンテキスト値として使用しま

す。CloudTrail ログエントリの requestParameters フィールドでは、暗号化コンテキストは次のようになります。

```
"encryptionContext": {  
  "aws:ebs:id": "vol-0cfb133e847d28be9"  
}
```

Amazon EC2 [CopySnapshot](#) オペレーションで作成された暗号化されたスナップショットの場合、Amazon EBS はスナップショット ID を暗号化コンテキスト値として使用します。CloudTrail ログエントリの requestParameters フィールドでは、暗号化コンテキストは次のようになります。

```
"encryptionContext": {  
  "aws:ebs:id": "snap-069a655b568de654f"  
}
```

Amazon EBS 障害の検出

暗号化された EBS ボリュームを作成するか、ボリュームを EC2 インスタンスにアタッチするには、Amazon EBS および Amazon EC2 インフラストラクチャで、EBS ボリュームの暗号化に指定した KMS キーを使用できる必要があります。KMS キーを使用できない場合 ([キーステータス](#)が Enabled ではない場合など)、ボリュームの作成またはボリュームのアタッチメントは失敗します。

この場合、Amazon EBS はイベントを Amazon EventBridge (旧 CloudWatch Events) に送信して、失敗について通知します。EventBridge では、これらのイベントに応じて自動アクションをトリガーするルールを設定できます。詳細については、「[Amazon EC2 ユーザーガイド](#)」の「Amazon EBS 向け Amazon CloudWatch Events」を参照してください。特に以下のセクションが重要となります。

- [ボリュームのアタッチ時または再アタッチ時の無効な暗号化キー](#)
- [ボリューム作成時の無効な暗号化キー](#)

これらの障害を修正するには、EBS ボリューム暗号化のために指定した KMS キーが有効になっていることを確認します。これを行うには、まず [KMS キーを表示](#)して、現在のキーの状態 (の Status 列 AWS Management Console) を確認します。次に、以下のリンクのいずれかで情報を確認します。

- KMS キーのキーステータスが無効になっている場合は、[有効](#)にします。
- KMS キーのキーステータスがインポート保留中になっている場合は、[キーマテリアルをインポート](#)します。

- KMS キーのキーステータスが削除保留中になっている場合は、[キーの削除をキャンセル](#)します。

AWS CloudFormation を使用して暗号化された Amazon EBS ボリュームを作成する

[AWS CloudFormation](#) を使用して、暗号化された Amazon EBS ボリュームを作成できます。詳細については、AWS CloudFormation ユーザーガイドの [AWS::EC2::Volume](#) を参照してください。

Amazon EMR が を使用する方法 AWS KMS

[Amazon EMR](#) クラスターを使用する場合、永続的ストレージの場所に保存する前に、保管中のデータを暗号化するようにクラスターを設定できます。保存データは、EMR ファイルシステム (EMRFS) かクラスターノードのストレージボリューム、またはその両方で暗号化できます。保管中のデータを暗号化するには、AWS KMS keyを使用します。以下のトピックでは、Amazon EMR クラスターが KMS キーを使用して保管中のデータを暗号化する方法について説明します。

Important

Amazon EMR は、[対称 KMS キー](#)のみをサポートします。[非対称 KMS キー](#)を使用して、Amazon EMR クラスター内の保管中のデータを暗号化することはできません。KMS キーが対称か非対称かを判断する方法については、「[さまざまなキータイプの特定](#)」を参照してください。

Amazon EMR クラスターは、転送中のデータも暗号化します。つまり、クラスターはネットワーク経由でデータを送信する前にデータを暗号化します。KMS キーを使用して送信中のデータを暗号化することはできません。詳細については、Amazon EMR 管理ガイドの[転送時のデータ暗号化](#)を参照してください。

Amazon EMR で使用できるすべての暗号化オプションの詳細については、Amazon EMR 管理ガイドの[暗号化オプション](#)を参照してください。

トピック

- [EMR ファイルシステム \(EMRFS\) のデータを暗号化する](#)
- [クラスターノードのストレージボリュームのデータを暗号化する](#)
- [暗号化コンテキスト](#)

EMR ファイルシステム (EMRFS) のデータを暗号化する

Amazon EMR クラスターは、次の 2 つの分散ファイルシステムを使用します。

- Hadoop Distributed File System (HDFS) HDFS 暗号化は、AWS KMSで KMS キーを使用しません。
- EMR ファイルシステム (EMRFS) EMRFS は HDFS の実装で、Amazon EMR クラスターが Amazon Simple Storage Service (Amazon S3) にデータを格納できるようにします。EMRFS は、4 種類の暗号化オプションをサポートしており、そのうち 2 種類は AWS KMSで KMS キーを使用します。EMRFS 暗号化オプションの 4 種類すべての詳細については、Amazon EMR 管理ガイドの[暗号化オプション](#)を参照してください。

KMS キーを使用する 2 種類の EMRFS 暗号化オプションは、Amazon S3 が提供する次の暗号化機能を使用します。

- [AWS Key Management Service \(SSE-KMS\) によるサーバー側の暗号化を使用したデータの保護](#)。Amazon EMR クラスターは、Simple Storage Service (Amazon S3) にデータを送信します。Amazon S3 は、KMS キーを使用してデータを暗号化してから、そのデータを S3 バケットに保存します。この仕組みについては、「[SSE-KMS を使用して EMRFS のデータを暗号化するプロセス](#)」を参照してください。
- [クライアント側の暗号化を使用したデータの保護 \(CSE-KMS\)](#)。Amazon EMR のデータは、AWS KMS key で暗号化されてから、ストレージのために Amazon S3 に送信されます。この仕組みについては、「[CSE-KMS を使用して EMRFS のデータを暗号化するプロセス](#)」を参照してください。

KMS を使用して EMRFS 上のデータを暗号化するように Amazon EMR クラスターを設定するときは、Amazon S3 または Amazon EMR クラスターで使用する KMS キーを選択します。SSE-KMS を使用して、エイリアス aws/s3 を持つ Amazon S3 の AWS マネージドキー、または作成する対称カスタマーマネージドキーを選択できます。クライアント側の暗号化では、作成する対称カスタマーマネージドキーを選択する必要があります。カスタマーマネージドキーを選択する際に、Amazon EMR クラスターに KMS キーの使用許可があることを確認する必要があります。詳細については、「Amazon EMR 管理ガイド」の[「暗号化 AWS KMS keys に使用する」](#)を参照してください。

サーバー側の暗号化とクライアント側の暗号化のどちらの場合でも、選択する KMS キーが[エンベロープ暗号化](#)ワークフローのルートキーになります。データは、KMS [キーで暗号化された一意のデータキー](#)で暗号化されます AWS KMS。暗号化されたデータとその暗号化されたデータキーの

ピーは、1 つの暗号化オブジェクトとして S3 バケットと一緒に保存されます。この仕組みについては、次のトピックを参照してください。

トピック

- [SSE-KMS を使用して EMRFS のデータを暗号化するプロセス](#)
- [CSE-KMS を使用して EMRFS のデータを暗号化するプロセス](#)

SSE-KMS を使用して EMRFS のデータを暗号化するプロセス

SSE-KMS を使用するように Amazon EMR クラスターを設定すると、暗号化プロセスは次のように動作します。

1. クラスターは、S3 バケットに格納するために Amazon S3 にデータを送信します。
2. Amazon S3 は [GenerateDataKey](#) リクエストを に送信し AWS KMS、SSE-KMS を使用するようにクラスターを設定したときに選択した KMS キーのキー ID を指定します。リクエストには暗号化コンテキストが含まれます。詳細については、「[暗号化コンテキスト](#)」を参照してください。
3. AWS KMS は一意のデータ暗号化キー (データキー) を生成し、このデータキーの 2 つのコピーを Amazon S3 に送信します。コピーのうち一方は暗号化されない形式 (プレーンテキスト) で、もう一方は KMS キーで暗号化されます。
4. Amazon S3 は、プレーンテキストデータキーを使用してステップ 1 で受信したデータを暗号化し、使用後できるだけ早くプレーンテキストデータキーをメモリから削除します。
5. Amazon S3 は、暗号化されたデータとデータキーの暗号化されたコピーを、1 つの暗号化されたオブジェクトとして S3 バケットに格納します。

この復号プロセスは、次のように行われます。

1. クラスターは、暗号化されたデータオブジェクトを S3 バケットへ要求します。
2. Amazon S3 は、暗号化されたデータキーを S3 オブジェクトから抽出し、暗号化されたデータキーを [Decrypt](#) リクエスト AWS KMS で に送信します。リクエストには[暗号化コンテキスト](#)が含まれます。
3. AWS KMS は、暗号化に使用したのと同じ KMS キーを使用して暗号化されたデータキーを復号し、復号された (プレーンテキスト) データキーを Amazon S3 に送信します。
4. Amazon S3 は、プレーンテキストデータキーを使用して暗号化されたデータを復号化し、使用後できるだけ早くプレーンテキストデータキーをメモリから削除します。
5. Amazon S3 は、復号化されたデータをクラスターに送信します。

CSE-KMS を使用して EMRFS のデータを暗号化するプロセス

CSE-KMS を使用するように Amazon EMR クラスターを設定すると、暗号化プロセスは次のように動作します。

1. Amazon S3 にデータを保存する準備ができたなら、クラスターは [GenerateDataKey](#) リクエストを送信し AWS KMS、CSE-KMS を使用するようにクラスターを設定したときに選択した KMS キーのキー ID を指定します。リクエストには暗号化コンテキストが含まれます。詳細については、「[暗号化コンテキスト](#)」を参照してください。
2. AWS KMS は一意のデータ暗号化キー (データキー) を生成し、このデータキーの 2 つのコピーをクラスターに送信します。コピーのうち一方は暗号化されない形式 (プレーンテキスト) で、もう一方は KMS キーで暗号化されます。
3. クラスターは、プレーンテキストデータキーを使用してデータを暗号化し、使用後できるだけ早くそのプレーンテキストデータキーをメモリから削除します。
4. クラスターは、暗号化データと暗号化されたデータキーのコピーを 1 つの暗号化オブジェクトにまとめます。
5. クラスターは、暗号化されたオブジェクトを Amazon S3 に送信してストレージします。

この復号プロセスは、次のように行われます。

1. クラスターは、暗号化されたデータオブジェクトを S3 バケットへ要求します。
2. Amazon S3 は、暗号化されたオブジェクトをクラスターに送信します。
3. クラスターは、暗号化されたオブジェクトから暗号化されたデータキーを抽出し、暗号化されたデータキーを [Decrypt](#) リクエスト AWS KMS で送信します。リクエストには [暗号化コンテキスト](#)が含まれます。
4. AWS KMS は、暗号化に使用したのと同じ KMS キーを使用して暗号化されたデータキーを復号し、復号された (プレーンテキスト) データキーをクラスターに送信します。
5. クラスターは、そのプレーンテキストデータキーを使用して、暗号化されたデータを復号し、使用後できるだけ早くプレーンテキストデータキーをメモリから削除します。

クラスターノードのストレージボリュームのデータを暗号化する

Amazon EMR クラスターは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの集合です。クラスター内のインスタンスはそれぞれ、クラスターノードまたはノードと呼ばれます。各ノードには、インスタンスストアボリュームと Amazon Elastic Block Store (Amazon EBS) ボリューム

の 2 種類のストレージボリュームがあります。これらのノードでストレージボリュームをいずれも暗号化するには、クラスターを構成して、[Linux Unified Key Setup \(LUKS\)](#) を使用できます (各ノードの起動ボリュームは不可)。これは、ローカルディスクの暗号化と呼ばれます。

クラスター向けにローカルディスクの暗号化を有効にすると、AWS KMS の KMS キーを使用して LUKS キーを暗号化できます。作成した[カスタマーマネージドキー](#)を選択する必要があります。[AWS マネージドキー](#)を使用することはできません。カスタマーマネージドキーを選択する場合は、Amazon EMR クラスターに KMS キーを使用するアクセス許可があることを確認する必要があります。詳細については、「Amazon EMR 管理ガイド」の「[暗号化 AWS KMS keys に使用する](#)」を参照してください。

KMS キーを使用してローカルディスクの暗号化を有効にする際、暗号化プロセスは次のようになります。

1. 各クラスターノードが起動すると、クラスターのローカルディスク暗号化を有効にしたときに選択した KMS キーのキー ID を指定して AWS KMS、[GenerateDataKey](#) リクエストが に送信されます。
2. AWS KMS は一意のデータ暗号化キー (データキー) を生成し、このデータキーの 2 つのコピーをノードに送信します。コピーのうち一方は暗号化されない形式 (プレーンテキスト) で、もう一方は KMS キーで暗号化されます。
3. ノードでは、LUKS キーを保護するパスワードとして、base64 エンコードバージョンのプレーンテキストデータキーを使用します。ノードは、暗号化されたデータキーのコピーを起動ボリュームに保存します。
4. ノードが再起動すると、再起動されたノードは暗号化されたデータキーを [Decrypt](#) リクエスト AWS KMS とともに に送信します。
5. AWS KMS は、暗号化に使用したのと同じ KMS キーを使用して暗号化されたデータキーを復号し、復号された (プレーンテキスト) データキーをノードに送信します。
6. ノードでは、LUKS キーのロックを解除するパスワードとして、base64 エンコードバージョンのプレーンテキストデータキーを使用します。

暗号化コンテキスト

と統合された各 AWS サービスは AWS KMS、サービスが AWS KMS を使用してデータキーを生成したり、データを暗号化または復号したりするときに、暗号化[コンテキスト](#)を指定できます。暗号化コンテキストは、AWS KMS がデータの整合性をチェックするために使用する追加の認証済み情報です。サービスにおいて、暗号化オペレーションのために暗号化コンテキストを指定する際、復号

オペレーションと同じ暗号コンテキストを指定する必要があります。指定しない場合は復号できません。暗号化コンテキストも AWS CloudTrail ログファイルに書き込まれるため、特定の KMS キーが使用された理由を理解するのに役立ちます。

以下のセクションでは、KMS キーを使用する Amazon EMR 暗号化の各シナリオで使用される暗号化コンテキストについて説明します。

SSE-KMS による EMRFS 暗号化の暗号化コンテキスト

SSE-KMS を使用すると、Amazon EMR クラスターは Amazon S3 にデータを送信し、次に Amazon S3 が KMS キーを使用してデータを暗号化してから S3 バケットに保存します。この場合、Amazon S3 は、送信先の各 [GenerateDataKey](#) および [Decrypt](#) リクエストで、S3 オブジェクトの Amazon リソースネーム (ARN) を暗号化コンテキストとして使用します AWS KMS。次の例は、Amazon S3 が使用する暗号化コンテキストの JSON 表現を示しています。

```
{ "aws:s3:arn" : "arn:aws:s3:::S3_bucket_name/S3_object_key" }
```

CSE-KMS による EMRFS 暗号化の暗号化コンテキスト

CSE-KMS では、Amazon EMR クラスターは KMS キーを使用してデータを暗号化してから Amazon S3 に送信して保存します。この場合、クラスターは、送信する各 [GenerateDataKey](#) および [Decrypt](#) リクエストの暗号化コンテキストとして KMS キーの Amazon リソースネーム (ARN) を使用します AWS KMS。次の例では、クラスターが使用する暗号化コンテキストの JSON 表現を示します。

```
{ "kms_cmek_id" : "arn:aws:kms:us-east-2:111122223333:key/0987ab65-43cd-21ef-09ab-87654321cdef" }
```

LUKS によるローカルディスク暗号化の暗号化コンテキスト

Amazon EMR クラスターが LUKS でローカルディスク暗号化を使用するとき、クラスターノードは AWS KMS に送信する [GenerateDataKey](#) および [Decrypt](#) リクエストで暗号化コンテキストを指定しません。

Amazon Redshift が を使用する方法 AWS KMS

このトピックでは、Amazon Redshift が AWS KMS を使用してデータを暗号化する方法について説明します。

トピック

- [Amazon Redshift 暗号化](#)
- [暗号化コンテキスト](#)

Amazon Redshift 暗号化

Amazon Redshift データウェアハウスは、ノードと呼ばれるコンピューティングリソースの集合で、クラスターと呼ばれるグループに編成されています。各クラスターは Amazon Redshift エンジンを実行し、1 つ以上のデータベースを含みます。

Amazon Redshift は、暗号化に 4 階層のキーベースのアーキテクチャを使用します。アーキテクチャは、データ暗号化キー、データベースキー、クラスターキー、ルートキーで構成されます。をルートキー AWS KMS key として使用できます。

データ暗号化キーは、クラスター内のデータブロックを暗号化します。各データブロックに、ランダムに生成された AES-256 キーが割り当てられます。これらのキーは、クラスターのデータベースキーを使用して暗号化されます。

データベースキーは、クラスターのデータ暗号化キーを暗号化します。データベースキーは、ランダムに生成された AES-256 キーです。これは Amazon Redshift クラスターとは別のネットワークのディスクに保存され、安全なチャネルを介してクラスターに渡されます。

クラスターキーは、Amazon Redshift クラスターのデータベースキーを暗号化します。AWS KMS、AWS CloudHSM、または外部ハードウェアセキュリティモジュール (HSM) を使用して、クラスターキーを管理できます。詳細については、[Amazon Redshift データベース暗号化](#) のドキュメントを参照してください。

暗号化をリクエストするには、Amazon Redshift コンソールで適切なチェックボックスをオンにします。暗号化ボックスの下に表示されるリストから[カスタマーマネージドキー](#)を 1 つ選択して、指定できます。カスタマーマネージドキーを指定しない場合、Amazon Redshift はアカウントで Amazon Redshift の [AWS マネージドキー](#) を使用します。

Important

Amazon Redshift は、対称暗号化 KMS キーのみをサポートします。Amazon Redshift 暗号化ワークフローでは、非対称 KMS キーは使用できません。KMS キーが対称か非対称かを判断する方法については、[さまざまなキータイプの特定](#) を参照してください。

暗号化コンテキスト

と統合されている各サービスは、データキーのリクエスト、暗号化、復号時に暗号化[コンテキスト](#) AWS KMS を指定します。暗号化コンテキストは、AWS KMS がデータの整合性をチェックするために使用する追加の認証データ (AAD) です。つまり、暗号化オペレーションで暗号化コンテキストを指定すると、復号オペレーションでもそのコンテキストが指定され、指定しなかった場合、復号は成功しません。Amazon Redshift は、暗号化コンテキストにクラスター ID と作成時間を使用します。CloudTrail ログファイルの `requestParameters` フィールドでは、暗号化コンテキストは次のようになります。

```
"encryptionContext": {
  "aws:redshift:arn": "arn:aws:redshift:region:account_ID:cluster:cluster_name",
  "aws:redshift:createtime": "20150206T1832Z"
},
```

CloudTrail ログでクラスター名を検索して、AWS KMS key (KMS キー) を使用して実行されたオペレーションを確認できます。このオペレーションには、クラスターの暗号化、クラスターの復号、およびデータキーの生成が含まれます。

AWS KMS リファレンス

次のリファレンス資料は、KMS キーの使用と管理に関する有用な情報を提供します。

- [キータイプリファレンス](#)。各 AWS KMS API オペレーションをサポートする KMS キーのタイプを一覧表示します。

質問例: KMS キーを署名する RSA を有効または無効にすることはできますか？

- [キーステータス表](#)。KMS キーのキーステータスが、AWS KMS API オペレーションでの使用にどのように影響するかを示します。

質問例: 削除が保留中である KMS キーのエイリアスを変更することはできますか？

- [AWS KMS API アクセス許可のリファレンス](#)。各 AWS KMS API オペレーションに必要なアクセス許可に関する情報を提供します。

検索するには: 別の AWS アカウントのキーで [GetKeyPolicy](#) を実行できますか？ IAM ポリシーで kms:Decrypt アクセス許可を付与できますか。

- [ViaService リファレンス](#)。kms:ViaService 条件キーをサポートする AWS サービス一覧表示します。

質問例: Amazon ElastiCache からの場合のみアクセス許可を付与する kms:ViaService 条件キーを使用できますか？ Amazon Neptune の場合はどうですか。

- [AWS KMS の料金](#)。KMS キーの価格を一覧表示して説明します。

質問例: 非対称キーの使用にかかる費用はどれくらいですか？

- [AWS KMS リクエストクォータ](#)。各アカウントとリージョンの AWS KMS API リクエストの 1 秒あたりのクォータを一覧表示します。

質問例: [Decrypt](#) リクエストを毎秒いくつ実行できますか？ カスタムキーストアの KMS キーで、いくつの [Decrypt](#) リクエストを実行できますか。

- [AWS KMS リソースクォータ](#)。AWS KMS リソースのクォータを一覧表示します。

質問例: アカウントの各リージョンで何個の KMS キーを使用できますか？ 各 KMS キーに何個のエイリアスを使用できますか。

- [AWS と統合された サービス AWS KMS](#)。KMS キーを使用して作成、保存、管理するリソースを保護する AWS サービスを一覧表示します。

質問例: Amazon Connect は、Connect リソースを保護するのに KMS キーを使用しますか？

キーの AWS KMS キーステータス

に AWS KMS key は常にキー状態があります。KMS キーとその環境でのオペレーションは、キーの状態を変更できます。キーの状態は、一時的に変更することも、別のオペレーションによってキーの状態が変更されるまで変更することもできます。これらのオペレーションは、非同期的に、または API コールによって実行されます。

このセクションの表は、キーの状態が AWS KMS API オペレーションの呼び出しにどのように影響するかを示しています。キーステータスの結果として、KMS キーでのオペレーションは成功 (#)、失敗 (X)、または特定の条件でのみ成功する (?) と予想されます。多くの場合、この結果は KMS キーのインポートされたキーマテリアルによって異なります。

この表には、既存の KMS キーを使用する API オペレーションのみが含まれています。その他のオペレーション ([CreateKey](#) および [ListKeys](#) など) は省略されています。

トピック

- [キーステータスと KMS キーの種類](#)
- [キーステータスの表](#)

キーステータスと KMS キーの種類

KMS キーの種類によって、それが持つキーステータスが決まります。

- すべての KMS キーは、Enabled、Disabled、および PendingDeletion 状態になり得ます。
- ほとんどの KMS キーは Enabled ステータスで作成されます。インポートされたキーマテリアルを持つキーは PendingImport ステータスで作成されます。
- PendingImport ステータスは、[インポートされたキーマテリアル](#)を持つ KMS キーにのみ適用されます。インポートされたキーのキーマテリアルが削除されるか期限切れになると、状態は から Enabled に変わります PendingImport。
- Unavailable ステータスは、[カスタムキーストア](#)内の KMS キーにのみ適用されます。[AWS CloudHSM キーストア](#)の KMS キーは、カスタムキーストアが AWS CloudHSM クラスタから意図的に切断 Unavailable されている場合です。カスタムキーストアがその[外部キーストアプロキシ](#)から意図的に切断された場合、[外部キーストア](#)の KMS キーは Unavailable です。使用できな

い KMS キーを表示および管理することはできますが、暗号化オペレーションで使用することはできません。

カスタムキーストア内の KMS キーのキーの状態は、そのバックアップキーが変更されても影響を受けません。AWS CloudHSM キーストアの KMS キーは、AWS CloudHSM クラスター内の[関連するキーマテリアル](#)の変更の影響を受けません。外部キーストアの KMS キーは、外部のキーマネージャーの[外部キー](#)が変更されても影響を受けません。バックアップキーが無効化されるか削除されると、KMS キーの状態は変わりませんが、KMS キーを使用した暗号化オペレーションは失敗します。

- Creating、Updating、PendingReplicaDeletion のキーステータスは、[マルチリージョンキー](#)にのみ適用されます。
- マルチリージョンのレプリカキーは、作成中の一時的な Creating キーステータスです。[ReplicateKey](#) オペレーションの完了時、このプロセスはまだ進行中の可能性があります。レプリケートプロセスが完了すると、レプリカキーは Enabled または PendingImport ステータスになります。
- プライマリリージョンの更新中、マルチリージョンキーは一時的に Updating キーステータスになります。[UpdatePrimaryRegion](#) オペレーションの完了時、このプロセスはまだ進行中の可能性があります。更新プロセスが完了すると、プライマリキーとレプリカキーは、Enabled キーステータスを再開します。
- レプリカキーを持つマルチリージョンのプライマリキーの削除をスケジュールすると、プライマリキーはそのレプリカキーがすべて削除されるまで、PendingReplicaDeletion ステータスを保持します。その後、キーステータスが PendingDeletion に変わります。詳細については、「[Deleting multi-Region keys](#)」を参照してください。

キーステータスの表

次の表に、KMS キーのキーステータスが、AWS KMS オペレーションにどのような影響を与えるかを示します。

番号付きの脚注の説明 ([n]) は、このトピックの最後にあります。

Note

この表のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

API	有効	無効	削除保留中 レプリカの削除保留中	インポートの保留中	使用不可	作成	[更新中]
CancelKey Deletion	 [4]	 [4]		 [4]	 [4], [13]	 [4]	 [4]
CreateAlias			 [3]				
CreateGrant		 [1]	 [2] または [3]	 [5]		 [14]	
Decrypt		 [1]	 [2] または [3]	 [5]	 [11]	 [14]	
DeleteAlias							
DeleteImportedKeyMaterial	 [9]	 [9]	 [9]		該当なし	 [14]	 [15]
DeriveSharedSecret		 [1]		 [5]	該当なし	 [14]	

API	有効	無効	削除保留 中 レプリカ の削除保 留中	インポー トの保留 中	使用不可	作成	[更新中]
			[2] また は [3]				
DescribeKey	✓	✓	✓	✓	✓	✓	✓
DisableKey	✓	✓	✗ [3]	✗ [5]	✓ [12]	✗ [14]	✗ [15]
DisableKeyRotation	❓ [7]	✗ [1] また は [7]	✗ [3] また は [7]	✗ [6]	✗ [7]	✗ [14]	❓ [7]
EnableKey	✓	✓	✗ [3]	✗ [5]	✓ [12]	✗ [14]	✗ [15]
EnableKeyRotation	❓ [7]	✗ [1] また は [7]	✗ [3] また は [7]	✗ [6]	✗ [7]	✗ [14]	❓ [7]
暗号化	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	✗ [11]	✗ [14]	✓

API	有効	無効	削除保留 中 レプリカ の削除保 留中	インポー トの保留 中	使用不可	作成	[更新中]
GenerateDataKey	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	✗ [11]	✗ [14]	✓
GenerateDataKeyPair	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	✗ [7]	✗ [14]	✓
GenerateDataKeyPairWithoutPlaintext	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	✗ [7]	✗ [14]	✓
GenerateDataKeyWithoutPlaintext	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	✗ [11]	✗ [14]	✓
GenerateMac	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	該当なし	✗ [14]	✓
GetKeyPolicy	✓	✓	✓	✓	✓	✓	✓

API	有効	無効	削除保留 中 レプリカ の削除保 留中	インポー トの保留 中	使用不可	作成	[更新中]
GetKeyRotationStatus	 [7]	 [7]	 [7]	 [6]	 [7]	 [7]	 [7]
GetParametersForImport	 [9]	 [9]	 [8] また は [9]		 [9]	 [14]	 [15]
GetPublicKey	 [9]	 [9]	 [2] また は [3]		該当なし	 [14]	
ImportKeyMaterial	 [9]	 [9]	 [9]		 [9]	 [14]	
ListAliases	 [9]	 [9]	 [9]	 [9]	 [9]	 [9]	 [9]
ListGrants	 [9]	 [9]	 [9]	 [9]	 [9]	 [9]	 [9]
ListKeyPolicies	 [9]	 [9]	 [9]	 [9]	 [9]	 [9]	 [9]

API	有効	無効	削除保留 中 レプリカ の削除保 留中	インポ ートの保 留 中	使用不可	作成	[更新中]
ListKeyRo tations	 [7]	 [7]	 [7]	 [6]	 [7]	 [7]	 [7]
ListResou rceTags							
PutKeyPol icy							
ReEncrypt		 [1]	 [2] また は [3]	 [5]	 [11]	 [14]	
Replicate Key		 [1]	 [2] また は [3]	 [5]	該当なし	 [14]	 [15]
RetireGra nt							
RevokeGra nt							
RotateKey OnDemand	 [7]	 [1] また は [7]	 [3] また は [7]	 [5]	 [7]	 [14]	 [7]

API	有効	無効	削除保留 中 レプリカ の削除保 留中	インポー トの保留 中	使用不可	作成	[更新中]
ScheduleKeyDeletion	✓	✓	 [3]	✓	✓	✓	 [15]
Sign	✓	 [1]	 [2] また は [3]	 [5]	該当なし	 [14]	✓
TagResource	✓	✓	 [3]	✓	✓	✓	✓
UntagResource	✓	✓	 [3]	✓	✓	✓	✓
UpdateAlias	✓	✓	 [10]	✓	✓	✓	✓
UpdateKeyDescription	✓	✓	 [3]	✓	✓	✓	✓

API	有効	無効	削除保留 中 レプリカ の削除保 留中	インポ ートの保 留 中	使用不可	作成	[更新中]
UpdatePrimaryRegion	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	該当なし	✗ [14]	✓
検証	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	該当なし	✗ [14]	✓
VerifyMac	✓	✗ [1]	✗ [2] また は [3]	✗ [5]	該当なし	✗ [14]	✓

テーブルの詳細

- [1] DisabledException: `<key ARN>` is disabled.
- [2] DisabledException: `<key ARN>` is pending deletion (or pending replica deletion).
- [3] KMSInvalidStateException: `<key ARN>` is pending deletion (or pending replica deletion).
- [4] KMSInvalidStateException: `<key ARN>` is not pending deletion (or pending replica deletion).
- [5] KMSInvalidStateException: `<key ARN>` is pending import because no key material has ever been imported or one of the imported key materials is deleted or expired.

- [6] `UnsupportedOperationException`: `<key ARN>` origin is EXTERNAL which is not valid for this operation.
- [7] KMS キーがカスタムキーストアにある場合: `UnsupportedOperationException`。
- [8] KMS キーがインポートされたキーマテリアルを持つ場合: `KMSInvalidStateException`
- [9] KMS キーにインポートされたキーマテリアルがない場合:
`UnsupportedOperationException`。
- [10] ソース KMS キーが削除保留中の場合、コマンドは成功します。送信先の KMS キーが削除保留中の場合、コマンドは次のエラーで失敗します: `KMSInvalidStateException` : `<key ARN>` is pending deletion.
- [11] `KMSInvalidStateException`: `<key ARN>` is unavailable. 使用不可能な KMS キーでこのオペレーションを実行することはできません。
- [12] オペレーションは成功しますが、KMS キーのキーステータスは、使用可能になるまで変更されません。
- [13] カスタムキーストアの KMS キーが削除保留中の場合、KMS キーが使用不可能になっても、そのキーステータスは `PendingDeletion` のままになります。これにより、待機期間中はいつでも KMS キーの削除をキャンセルできます。
- [14] は、マルチリージョンキー () のレプリケート中にこの例外
を `KMSInvalidStateException`: `<key ARN>` is creating. AWS KMS スローします `ReplicateKey`。
- [15] は、マルチリージョンキー () のプライマリリージョンの更新中にこの例外
を `KMSInvalidStateException`: `<key ARN>` is updating. AWS KMS スローします `UpdatePrimaryRegion`。

キータイプリファレンス

AWS KMS は、さまざまなタイプの KMS キーに対してさまざまな機能をサポートしています。例えば、[対称データキー](#)と[非対称データキーペア](#)の生成には、[対称暗号化 KMS キー](#)のみを使用できます。また、[キーマテリアルのインポート](#)と[自動キーローテーション](#)は対称暗号化 KMS キーのみでサポートされ、[カスタムキーストア](#)では対称暗号化 KMS キーのみを作成できます。

このリファレンスには 2 つの表が含まれています。

- [キータイプテーブル](#)には、対称暗号化 KMS キー、非対称 KMS キー、および HMAC KMS キーに有効な AWS KMS オペレーションが一覧表示されます。

- [特殊な機能の表](#)には、マルチリージョン KMS キー、インポートしたキーマテリアルを含む KMS キー、カスタムキーストアの KMS キーで有効な AWS KMS オペレーションが記載されています。

キータイプの表

この表のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

AWS KMS API オペレーション	対称暗号化 KMS キー	HMAC KMS キー	非対称 KMS キー (ENCRYPT/DECRYPT)	非対称 KMS キー (SIGN_VERIFY)	非対称 KMS キー (KEY_AGREEMENT)
CancelKeyDeletion	はい	あり	あり	あり	はい
CreateAlias	はい	あり	あり	あり	はい
CreateGrant	はい	あり	あり	あり	はい
CreateKey	はい	あり	あり	あり	はい
Decrypt	はい	なし	あり	いいえ	いいえ
DeleteAlias	はい	あり	あり	あり	はい
DeleteImportedKeyMaterial インポートしたキーマテリアルを含む KMS キーのみで有効です (Origin は EXTERNAL)。	はい	あり	あり	あり	はい
DeriveSharedSecret	いいえ	いいえ	いいえ	いいえ	はい
DescribeKey	はい	あり	あり	あり	はい

AWS KMS API オペレーション	対称暗号化 KMS キー	HMAC KMS キー	非対称 KMS キー (ENCRYPT/DECRYPT)	非対称 KMS キー (SIGN_VERIFY)	非対称 KMS キー (KEY_AGREEMENT)
DisableKey	はい	あり	あり	あり	はい
DisableKeyRotation	はい キーマテリアルを持つ KMS AWS KMS キーでのみ有効です (Origin は)AWS_k	いいえ	いいえ	いいえ	いいえ
EnableKey	はい	あり	あり	あり	はい
EnableKeyRotation	はい キーマテリアルを持つ KMS AWS KMS キーでのみ有効です (Origin は)AWS_k	いいえ	いいえ	いいえ	いいえ

AWS KMS API オペレーション	対称暗号化 KMS キー	HMAC KMS キー	非対称 KMS キー (ENCRYPT/DECRYPT)	非対称 KMS キー (SIGN_VERIFY)	非対称 KMS キー (KEY_AGREEMENT)
暗号化	はい	なし	あり	いいえ	いいえ
GenerateDataKey	はい	いいえ	いいえ	いいえ	いいえ
GenerateDataKeyPair 対称暗号化 KMS キーによって保護される非対称データキーペアを生成します。	はい カスタムキーストアの KMS キーでは有効ではありません。	いいえ	いいえ	いいえ	いいえ
GenerateDataKeyPairWithoutPlaintext 対称暗号化 KMS キーによって保護される非対称データキーペアを生成します。	はい カスタムキーストアの KMS キーでは有効ではありません。	いいえ	いいえ	いいえ	いいえ
GenerateDataKeyWithoutPlaintext	はい	いいえ	いいえ	いいえ	いいえ
GenerateMac	いいえ	あり	いいえ	いいえ	いいえ

AWS KMS API オペレーション	対称暗号化 KMS キー	HMAC KMS キー	非対称 KMS キー (ENCRYPT/DECRYPT)	非対称 KMS キー (SIGN_VERIFY)	非対称 KMS キー (KEY_AGREEMENT)
GetKeyPolicy	はい	あり	あり	あり	はい
GetKeyRotationStatus	はい	はい (KeyRotationEnabled は常に false になります。)	はい (KeyRotationEnabled は常に false になります。)	はい (KeyRotationEnabled は常に false になります。)	はい (KeyRotationEnabled は常に false になります。)
GetParametersForImport インポートしたキーマテリアルを含む KMS キーのみで有効です (Origin は EXTERNAL)。	はい	あり	あり	あり	はい
GetPublicKey	いいえ	いいえ	はい	あり	はい
ImportKeyMaterial インポートしたキーマテリアルを含む KMS キーのみで有効です (Origin は EXTERNAL)。	はい	あり	あり	あり	はい

AWS KMS API オペレーション	対称暗号化 KMS キー	HMAC KMS キー	非対称 KMS キー (ENCRYPT/DECRYPT)	非対称 KMS キー (SIGN_VERIFY)	非対称 KMS キー (KEY_AGREEMENT)
ListAliases	はい	あり	あり	あり	はい
ListGrants	はい	あり	あり	あり	はい
ListKeyPolicies	はい	あり	あり	あり	はい
ListKeyRotations	はい	はい (Rotations フィールドは常に NULL または空です。)	はい (Rotations フィールドは常に NULL または空です。)	はい (Rotations フィールドは常に NULL または空です。)	はい (Rotations フィールドは常に NULL または空です。)
ListResourceTags	はい	あり	あり	あり	はい
ListRetirableGrants	はい	あり	あり	あり	はい
PutKeyPolicy	はい	あり	あり	あり	はい
ReEncrypt	はい	なし	あり	いいえ	いいえ
ReplicateKey – マルチリージョンキーでのみ有効	はい	あり	あり	あり	はい
RetireGrant	はい	あり	あり	あり	はい
RevokeGrant	はい	あり	あり	あり	はい

AWS KMS API オペレーション	対称暗号化 KMS キー	HMAC KMS キー	非対称 KMS キー (ENCRYPT/DECRYPT)	非対称 KMS キー (SIGN_VERIFY)	非対称 KMS キー (KEY_AGREEMENT)
RotateKeyOnDemand	はい AWS_KMS または オEXTERN ジンを持つカスタマー管理の対称暗号化 KMS キーでのみ有効です。	いいえ	いいえ	いいえ	いいえ
ScheduleKeyDeletion	はい	あり	あり	あり	はい
Sign	いいえ	いいえ	いいえ	あり	いいえ
TagResource	あり	あり	あり	あり	はい
UntagResource	あり	あり	あり	あり	はい

AWS KMS API オペレーション	対称暗号化 KMS キー	HMAC KMS キー	非対称 KMS キー (ENCRYPT/DECRYPT)	非対称 KMS キー (SIGN_VERIFY)	非対称 KMS キー (KEY_AGREEMENT)
UpdateAlias	はい	あり	あり	あり	はい
現在の KMS キーと新しい KMS キーは同じタイプでなければならず (両方とも対称か両方とも非対称、もしくは両方とも HMAC)、 キーの用途 が同じでなければなりません。					
UpdateKeyDescription	はい	あり	あり	あり	はい
UpdateReplicaRegion	はい	あり	あり	あり	はい
– マルチリージョンキーでのみ有効					
検証	いいえ	いいえ	いいえ	あり	いいえ
VerifyMac	いいえ	あり	いいえ	いいえ	いいえ

特殊な機能の表

この表は、各タイプの専用キーでサポートされている AWS KMS API オペレーションを示しています。

この表を見るときは、次の相互関係に注意します。

- [マルチリージョンキー](#):

- マルチリージョンキーは、対称暗号化 KMS キー、非対称 KMS キー、HMAC KMS キー、インポートされたキーマテリアルを含む KMS キーのいずれかです。
- カスタムキーストアでマルチリージョンキーを作成することはできません。

- [インポートされたキーマテリアル](#)

- 対称暗号化 KMS キー、非対称 KMS キー、HMAC KMS キーのキーマテリアルをインポートできます。
- [インポートしたキーマテリアルのマルチリージョンキー](#)を作成できます。
- カスタムキーストアでは、インポートしたキーマテリアルを持つキーは作成できません。
- 自動キーローテーション (EnableKeyRotation、DisableKeyRotation) は、インポートされたキーマテリアルを持つ KMS キーではサポートされません。
- オンデマンドキーローテーション (RotateKeyOnDemand) は、インポートされたキーマテリアルを持つ単一リージョンの対称暗号化 KMS キーでサポートされています。

- [カスタムキーストア](#)

- カスタムキーストアは、対称暗号化 KMS キーのみをサポートします。
- 非対称キーペア (GenerateDataKeyPair、GenerateDataKeyPairWithoutPlaintext) の対称オペレーションは、カスタムキーストアの KMS キーではサポートされていません。
- 自動キー回転機能 (EnableKeyRotation、DisableKeyRotation) は、カスタムキーストアの KMS キーではサポートされていません。
- カスタムキーストアでマルチリージョンキーを作成することはできません。

この表のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
CancelKeyDeletion	✓	✓	✓
CreateAlias	✓	✓	✓
CreateGrant	✓	✓	✓

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
CreateKey マルチリージョンプライマリキー、キーマテリアルがインポートされた KMS キー、カスタムキーストアの KMS キーのいずれかを作成するときは <code>CreateKey</code> を使用できます。マルチリージョンのレプリカキーを作成するには、 <code>ReplicateKey</code> リソースを使用します。	✓	✓	✓
Decrypt KeyUsage が <code>ENCRYPT_D</code> <code>ECRYPT</code> の場合のみ有効です。	✓	✓	✓
DeleteAlias	✓	✓	✓
DeleteImportedKeyMaterial インポートされたキーマテリアルを含むキーのみで有効です (Origin は <code>EXTERNAL</code>)。	✓	✓	✗

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
DeriveSharedSecret	 KeyUsage が の場合にのみ有効です (KEY_AGREEMENT)	 KeyUsage が の場合にのみ有効です (KEY_AGREEMENT)	
DescribeKey			
DisableKey			
DisableKeyRotation	 キーマテリアルを持つ対称暗号化 AWS KMS キーでのみ有効です (Origin は)AWS_KMS。		
EnableKey	 対称暗号化 KMS キーのみで有効です。		

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
EnableKeyRotation	 キーマテリアルを持つ対称暗号化 AWS KMS キーでのみ有効です (Origin は)AWS_KMS。		
暗号化	 KeyUsage が ENCRYPT_D ECRYPT の場合のみ有効です。		
GenerateDataKey	 対称暗号化 KMS キーのみで有効です。		
GenerateDataKeyPair	 対称暗号化 KMS キーのみで有効です。		

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
GenerateDataKeyPairWithoutPlaintext	 対称暗号化 KMS キーのみで有効です。		
GenerateDataKeyWithoutPlaintext	 対称暗号化 KMS キーのみで有効です。		
GenerateMac HMAC KMS キーのみで有効です。			
GetKeyPolicy			
GetKeyRotationStatus		 (KeyRotationEnabled は常に false になります。)	

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
GetParametersForImport	 インポートされたキーマテリアルを含むキーのみで有効です (Origin は EXTERNAL)。		
GetPublicKey 非対称 KMS キーのみで有効です。			
ImportKeyMaterial	 インポートされたキーマテリアルを含むキーのみで有効です (Origin は EXTERNAL)。		
ListAliases			
ListGrants			
ListKeyPolicies			

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
ListKeyRotations	 AWS_KMS オリジンを持つ対称暗号化キーでのみ有効です。	 単一リージョンの対称暗号化キーでのみ有効です。	
ListResourceTags			
ListRetirableGrants			
PutKeyPolicy			
ReEncrypt	 KeyUsage が ENCRYPT_D ECRYPT の場合のみ有効です。		
ReplicateKey	 マルチリージョンプライマリキーのみで有効です。	 マルチリージョンプライマリキーのみで有効です。	
RetireGrant			
RevokeGrant			

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
RotateKeyOnDemand	 AWS_KMS オリジンを持つ対称暗号化キーでのみ有効です。	 単一リージョンの対称暗号化キーでのみ有効です。	
ScheduleKeyDeletion			
Sign KeyUsage が SIGN_VERIFY の場合のみ有効です。			
TagResource			
UntagResource			
UpdateAlias - 現在の KMS キーと新しい KMS キーは同じタイプでなければならず (両方とも対称か両方とも非対称、もしくは両方とも HMAC)、 キーの用途 が同じでなければなりません。			
UpdateKeyDescription			

AWS KMS API オペレーション	マルチリージョンキー	インポートされたキーマテリアル	カスタムキーストアの KMS キー
UpdateReplicaRegion	✓	✓	✗
		マルチリージョンキーでのみ有効です。	
検証	✓	✓	✗
KeyUsage が SIGN_VERIFY の場合にのみ有効です。			
VerifyMac	✓	✓	✗
HMAC KMS キーのみで有効です。			

キー仕様のリファレンス

非対称 KMS キーまたは HMAC KMS キーを作成するときは、その[キーの仕様](#)を選択します。すべてのプロパティであるキー仕様は AWS KMS key、KMS キーの暗号化設定を表します。KMS キーの作成時に選択したキー仕様を変更することはできません。間違ったキー仕様を選択した場合は、[KMS キーを削除し](#)、新しいキー仕様を作成します。

Note

KMS キーのキー仕様は、「カスタマーマスターキー仕様」として知られていました。[CreateKey](#) オペレーションの CustomerMasterKeySpec パラメータは非推奨です。代わりに、KeySpec パラメータを使用します。CreateKey および [DescribeKey](#) オペレーションのレスポンスには、同じ値を持つ KeySpec および CustomerMasterKeySpec メンバーが含まれます。

キー仕様は、KMS キーが対称か非対称か、KMS キー内のキーマテリアルのタイプ、および が KMS キーに対して AWS KMS サポートする暗号化アルゴリズム、署名アルゴリズム、またはメッセージ

認証コード (MAC) アルゴリズムを決定します。選択するキー仕様は、通常、ユースケースと規制要件によって決まります。ただし、キー仕様の異なる KMS キーの暗号化オペレーションは料金が異なるため、クォータも異なる場合があります。料金の詳細については、「[AWS Key Management Service の料金](#)」を参照してください。リクエストクォータの詳細については、「[クォータのリクエスト](#)」を参照してください。

KMS キーの作成時に、プリンシパルが使用できるキー仕様を制限するには、[kms:KeySpec](#) 条件キーを使用します。kms:KeySpec 条件キーを使用して、プリンシパルが特定のキー仕様の KMS キーでのみ AWS KMS オペレーションを呼び出すことを許可することもできます。例えば、RSA_4096 キー仕様を持つ KMS キーの削除をスケジュールする許可を拒否できます。

AWS KMS では、KMS キーの次のキー仕様がサポートされています。

[対称暗号化キーの仕様](#) (デフォルト)

- SYMMETRIC_DEFAULT

[RSA キー仕様](#) (暗号化と復号、または署名と検証)

- RSA_2048
- RSA_3072
- RSA_4096

[楕円曲線のキー仕様](#)

- 非対称 NIST 推奨[楕円曲線キーペア](#) (署名と検証、または共有シークレットの取得のいずれか)
 - ECC_NIST_P256 (secp256r1)
 - ECC_NIST_P384 (secp384r1)
 - ECC_NIST_P521 (secp521r1)
- その他の非対称楕円曲線キーペア (署名と検証)
 - ECC_SECG_P256K1 ([secp256k1](#))、暗号化に一般的に使用されます。

[SM2 キー仕様](#) (暗号化と復号、または署名と検証、または共有シークレットの取得のいずれか)

- SM2 (中国リージョンのみ)

[HMAC キーの仕様](#)

- HMAC_224
- HMAC_256
- HMAC_384
- HMAC_512

ML-DSA キー仕様

- ML_DSA_44
- ML_DSA_65
- ML_DSA_87

SYMMETRIC_DEFAULT キー仕様

デフォルトのキー仕様である SYMMETRIC_DEFAULT は、対称暗号化 KMS キーのキー仕様です。AWS KMS コンソールで対称キータイプと暗号化および復号キーの使用を選択すると、SYMMETRIC_DEFAULT キー仕様を選択されます。[CreateKey](#) オペレーションで、KeySpec 値を指定しない場合、SYMMETRIC_DEFAULT が選択されます。別のキー仕様を使用する理由がない場合は、SYMMETRIC_DEFAULT を選択することをお勧めします。

SYMMETRIC_DEFAULT は AES-256-GCM を表します。AES-256-GCM は [Galois Counter Mode \(GCM\)](#) の [アドバンスト暗号化スタンダード](#) (AES) に基づく対称アルゴリズムであり、安全な暗号化のための業界標準である 256 ビットキーを備えています。このアルゴリズムが生成する暗号文は、[暗号化コンテキスト](#)などの追加認証データ (AAD) をサポートし、GCM は暗号文での追加の整合性チェックを提供します。

AES-256-GCM で暗号化されたデータは、現在も将来も保護されています。暗号作成者は、このアルゴリズムには量子耐性があると考えています。理論上の将来、256 ビット AES-GCM キーで作成された暗号文に対する大規模な量子コンピューティング攻撃は、[キーの効果的なセキュリティを 128 ビットに低下させます](#)。ただし、このセキュリティレベルは、AWS KMS 暗号文に対するブルートフォース攻撃を実行不可能にするのに十分です。

中国リージョンでは唯一の例外で、SYMMETRIC_DEFAULT は SM4 暗号化を使用する 128 ビットの対称キーを表します。128 ビット SM4 キーは、中国リージョンでのみ作成できます。中国リージョンでは、256 ビット AES-GCM KMS キーを作成することはできません。

で対称暗号化 KMS キーを使用して、データを AWS KMS 暗号化、復号、再暗号化し、生成されたデータキーと統合されたデータキーペアを保護できます。AWS サービスは、対称暗号化 KMS キー AWS KMS を使用して保管中のデータを暗号化します。対称暗号化 KMS キーに[独自のキーマテリアルをインポート](#)し、[カスタムキーストア](#)で対称暗号化 KMS キーを作成することができます。対称および非対称 KMS キーで実行できるオペレーションを比較した表については、「[対称および非対称 KMS キーの比較](#)」を参照してください。

で対称暗号化 KMS キーを使用して、データを AWS KMS 暗号化、復号、再暗号化し、データキーとデータキーペアを生成できます。[マルチリージョン](#)の対称暗号化 KMS キーの作成、対称暗号化

KMS キーへの[独自のキーマテリアルのインポート](#)、および[カスタムキーストア](#)での対称暗号化 KMS キーの作成が可能です。異なるタイプの KMS キーで実行できるオペレーションを比較した表については、「[キータイプリファレンス](#)」を参照してください。

RSA キー仕様

RSA キー仕様を使用する場合、は RSA キーペアを使用して非対称 KMS キー AWS KMS を作成します。プライベートキーが暗号化 AWS KMS されていないままになることはありません。内でパブリックキーを使用するか AWS KMS、外部で使用するパブリックキーをダウンロードできます AWS KMS。

Warning

の外部でデータを暗号化する場合は AWS KMS、暗号化テキストを復号できることを確認してください。AWS KMS から削除された KMS キーのパブリックキー、署名と検証用に設定された KMS キーのパブリックキー、または KMS キーでサポートされていない暗号化アルゴリズムを使用する場合、データは回復できません。

では AWS KMS、非対称 KMS キーと RSA キーペアを使用して暗号化と復号、または署名と検証を行うことができますが、両方を使用することはできません。このプロパティはと呼ばれ[Key usage](#)、キー仕様とは別に決定されますが、キー仕様を選択する前に決定する必要があります。

AWS KMS は、暗号化と復号、または署名と検証のために次の RSA キー仕様をサポートしています。

- RSA_2048
- RSA_3072
- RSA_4096

RSA キー仕様は、RSA キーの長さ（ビット単位）によって異なります。選択する RSA キー仕様は、セキュリティ標準またはタスクの要件によって決定される場合があります。一般的に、タスクに実用的で手頃な価格の、最大のキーを使用します。RSA キー仕様の異なる KMS キーの暗号化オペレーションは、料金が異なります。AWS KMS 料金の詳細については、[AWS 「Key Management Service」の料金](#)」を参照してください。リクエストクォータの詳細については、「[クォータのリクエスト](#)」を参照してください。

暗号化および復号の RSA キー仕様

RSA 非対称 KMS キーを暗号化および復号に使用する場合は、パブリックキーで暗号化し、プライベートキーで復号します。RSA KMS キー AWS KMS の `Encrypt` オペレーションを呼び出すと、は RSA キーペアのパブリックキーと、指定した暗号化アルゴリズム AWS KMS を使用してデータを暗号化します。暗号文を復号するには、`Decrypt` オペレーションを呼び出し、同じ KMS キーと暗号化アルゴリズムを指定します。AWS KMS その後、は RSA キーペアのプライベートキーを使用してデータを復号します。

パブリックキーをダウンロードし、それを使用して外部でデータを暗号化することもできます AWS KMS。RSA KMS キーに対して が AWS KMS サポートする暗号化アルゴリズムを使用してください。暗号テキストを復号するには、同じ KMS キーと暗号化アルゴリズムを使用して `Decrypt` 関数を呼び出します。

AWS KMS は、RSA キー仕様の KMS キーに対して 2 つの暗号化アルゴリズムをサポートしています。[PKCS #1 v2.2](#) で定義されるこれらのアルゴリズムは、内部的に使用するハッシュ関数によって異なります。AWS KMS で、RSAES_OAEP アルゴリズムは、ハッシュ目的と [マスク生成関数](#) (MGF1) の両方に常に同じハッシュ関数を使用します。[Encrypt](#) および [Decrypt](#) オペレーションを呼び出すときは、暗号化アルゴリズムを指定する必要があります。リクエストごとに異なるアルゴリズムを選択できます。

RSA キー仕様にサポートされる暗号化アルゴリズム

暗号化アルゴリズム	アルゴリズムの説明
RSAES_OAEP_SHA_1	PKCS #1 v2.2、セクション 7.1。ハッシュと MGF1 マスク生成機能の両方に空のラベルとともに SHA-1 を使用した OAEP パディングによる RSA 暗号化。
RSAES_OAEP_SHA_256	PKCS #1、セクション 7.1。ハッシュと MGF1 マスク生成機能の両方に空のラベルとともに SHA-256 を使用した OAEP パディングによる RSA 暗号化。

特定の暗号化アルゴリズムを使用するように KMS キーを設定することはできません。ただし、[kms:EncryptionAlgorithm](#) ポリシー条件を使用して、プリンシパルが KMS キーで使用できる暗号化アルゴリズムを指定できます。

KMS キーの暗号化アルゴリズムを取得するには、AWS KMS コンソールで KMS キーの[暗号化設定を表示する](#)か、[DescribeKey](#) オペレーションを使用します。AWS KMS は、AWS KMS コンソールまたは [GetPublicKey](#) オペレーションを使用してパブリックキーをダウンロードするときに、キー仕様と暗号化アルゴリズムも提供します。

各リクエストで暗号化できるプレーンテキストデータの長さに基づいて、RSA キー仕様を選択できます。次の表に、[Encrypt](#) オペレーションに対する 1 回の呼び出しで暗号化できるプレーンテキストの最大サイズをバイト単位で示します。値は、キー仕様と暗号化アルゴリズムによって異なります。比較すると、対称暗号化 KMS キーは一度に最大 4,096 バイトを暗号化するために使用できます。

これらのアルゴリズムのプレーンテキストの最大長 (バイト単位) を計算するには、次の式を使用します。 $(key\_size\_in\_bits / 8) - (2 * hash\_length\_in\_bits / 8) - 2$ 。例えば、RSA_2048 と SHA-256 の場合、バイト単位のプレーンテキストの最大サイズは、 $(2048/8) - (2 * 256/8) - 2 = 190$ です。

暗号化オペレーションの最大プレーンテキストサイズ (バイト単位)

キー仕様	暗号化アルゴリズム	
	RSAES_OAEP_SHA_1	RSAES_OAEP_SHA_256
RSA_2048	214	190
RSA_3072	342	318
RSA_4096	470	446

署名および検証用の RSA キー仕様

署名と検証に RSA 非対称 KMS キーを使用する場合、プライベートキーを持つメッセージの署名を生成し、パブリックキーで署名を検証します。

非対称 KMS キー AWS KMS ので [Sign](#) オペレーションを呼び出すと、は RSA キーペアのプライベートキー、メッセージ、および指定した署名アルゴリズム AWS KMS を使用して署名を生成します。署名を検証するには、[Verify](#) オペレーションを呼び出します。署名に加えて、同じ KMS キー、メッセージ、署名アルゴリズムを指定します。AWS KMS その後、は RSA キーペアのパブリックキーを使用して署名を検証します。パブリックキーをダウンロードし、それを使用して外部の署名を検証することもできます AWS KMS。

AWS KMS は、RSA キー仕様のすべての KMS キーに対して次の署名アルゴリズムをサポートしています。[Sign](#) オペレーションと [Verify](#) オペレーションを呼び出すときは、署名アルゴリズムを指定する必要があります。リクエストごとに異なるアルゴリズムを選択できます。RSA キーペアで署名する場合は、RSASSA-PSS アルゴリズムが推奨されます。既存のアプリケーションとの互換性を保つため、RSASSA-PKCS1-v1_5 アルゴリズムが採用されています。

RSA キー仕様でサポートされる署名アルゴリズム

署名アルゴリズム	アルゴリズムの説明
RSASSA_PSS_SHA_256	PKCS #1 v2.2、セクション 8.1、メッセージダイジェストと MGF1 マスク生成機能の両方に SHA-256 とともに 256 ビットソルトを使用する PSS パディング付きの RSA 署名
RSASSA_PSS_SHA_384	PKCS #1 v2.2、セクション 8.1、メッセージダイジェストと MGF1 マスク生成機能の両方に SHA-384 とともに 384 ビットソルトを使用する PSS パディング付きの RSA 署名
RSASSA_PSS_SHA_512	PKCS #1 v2.2、セクション 8.1、メッセージダイジェストと MGF1 マスク生成機能の両方に SHA-512 とともに 512 ビットソルトを使用する PSS パディング付きの RSA 署名
RSASSA_PKCS1_V1_5_SHA_256	PKCS #1 v2.2、セクション 8.2、PKCS #1v1.5 パディングおよび SHA-256 を使用した RSA 署名
RSASSA_PKCS1_V1_5_SHA_384	PKCS #1 v2.2、セクション 8.2、PKCS #1v1.5 パディングおよび SHA-384 を使用した RSA 署名
RSASSA_PKCS1_V1_5_SHA_512	PKCS #1 v2.2、セクション 8.2、PKCS #1v1.5 パディングおよび SHA-512 を使用した RSA 署名

特定の署名アルゴリズムを使用するように KMS キーを設定することはできません。ただし、[km:SigningAlgorithm](#) ポリシー条件を使用して、プリンシパルが KMS キーで利用できる署名アルゴリズムを指定できます。

KMS キーの署名アルゴリズムを取得するには、AWS KMS コンソールまたは [DescribeKey](#) オペレーションを使用して KMS キーの[暗号化設定を表示します](#)。AWS KMS は、AWS KMS コンソールまたは [GetPublicKey](#) オペレーションを使用してパブリックキーをダウンロードするときに、キー仕様と署名アルゴリズムも提供します。

楕円曲線のキー仕様

楕円曲線 (ECC) キー仕様を使用する場合、は署名と検証、または共有シークレットの取得 (両方ではない) のために、ECC キーペアを持つ非対称 KMS キー AWS KMS を作成します。署名を生成もしくは共有シークレットを取得するプライベートキーでは、AWS KMS が暗号化されないままになることはありません。パブリックキーを使用して内部[の署名を検証する](#)か AWS KMS、外部で使用する[パブリックキーをダウンロードできます](#) AWS KMS。

AWS KMS は、非対称 KMS キーに対して次の ECC キー仕様をサポートしています。

- 非対称 NIST 推奨楕円曲線キーペア (署名と検証、または共有シークレットの取得のいずれか)
 - ECC_NIST_P256 (secp256r1)
 - ECC_NIST_P384 (secp384r1)
 - ECC_NIST_P521 (secp521r1)
- その他の非対称楕円曲線キーペア (署名と検証)
 - ECC_SECG_P256K1 ([secp256k1](#))。一般に暗号通貨に用いられる。

選択する ECC キー仕様は、セキュリティ標準またはタスクの要件によって決定される場合があります。一般的に、タスクに実用的で手頃な価格の、最も多くのポイントがある曲線を使用します。

非対称 KMS キーを作成して[共有シークレットを取得する](#)場合は、NIST が推奨する楕円曲線キー仕様のいずれかを使用します。共有シークレットの取得向けにサポートされている唯一のキーアグリーメントアルゴリズムは、[楕円曲線暗号化共同係数 Diffie-Hellman Primitive](#) (ECDH) です。共有シークレットをオフラインで取得する方法の例については、「[the section called “オフラインで共有シークレットを取得する”](#)」を参照してください。

暗号通貨で使用する非対称 KMS キーを作成する場合は、ECC_SECG_P256K1 キー仕様を使用します。このキー仕様を他の目的に使用することもできますが、Bitcoin やその他の暗号化通貨には必要です。

ECC キー仕様異なる KMS キー仕様は料金が異なるため、リクエストクォータも異なる場合があります。AWS KMS 料金の詳細については、[AWS Key Management Service 「の料金」](#)を参照してください。リクエストクォータの詳細については、「[クォータのリクエスト](#)」を参照してください。

次の表は、ECC キー仕様ごとに AWS KMS サポートする署名アルゴリズムを示しています。特定の署名アルゴリズムを使用するように KMS キーを設定することはできません。ただし、[km:SigningAlgorithm](#) ポリシー条件を使用して、プリンシパルが KMS キーで利用できる署名アルゴリズムを指定できます。

ECC キー仕様でサポートされる署名アルゴリズム

キー仕様	署名アルゴリズム	アルゴリズムの説明
ECC_NIST_P256	ECDSA_SHA_256	メッセージダイジェストのためにキーおよび SHA-256 で指定された曲線を使用する、NIST FIPS 186-4、セクション 6.4、ECDSA 署名。
ECC_NIST_P384	ECDSA_SHA_384	メッセージダイジェストのためにキーおよび SHA-384 で指定された曲線を使用する、NIST FIPS 186-4、セクション 6.4、ECDSA 署名。
ECC_NIST_P521	ECDSA_SHA_512	メッセージダイジェストのためにキーおよび SHA-512 で指定された曲線を使用する、NIST FIPS 186-4、セクション 6.4、ECDSA 署名。
ECC_SECG_P256K1	ECDSA_SHA_256	メッセージダイジェストのためにキーおよび SHA-256 で指定された曲線を使用する

キー仕様	署名アルゴリズム	アルゴリズムの説明
		る、NIST FIPS 186-4、セクション 6.4、ECDSA 署名。

HMAC KMS キーの主な仕様

AWS KMS は、さまざまな長さの対称 HMAC キーをサポートしています。選択するキー仕様は、セキュリティ、規制、またはビジネス要件に応じて異なります。キーの長さによって、[GenerateMac](#) および [VerifyMac](#) オペレーションで使用される MAC アルゴリズムが決まります。一般に、キーが長いほど安全性が高くなります。ユースケースにとって実用的な最も長いキーを使用してください。

HMAC キーの仕様	MAC アルゴリズム
HMAC_224	HMAC_SHA_224
HMAC_256	HMAC_SHA_256
HMAC_384	HMAC_SHA_384
HMAC_512	HMAC_SHA_512

ML-DSA キー仕様

ML-DSA キーは、量子後暗号化用に設計された Module-Lattice-Based Digital Signature Algorithm (ML-DSA) で使用される暗号化キーです。このアルゴリズムは、NIST (米国国立標準技術研究所) の標準化の取り組みの一部であり、特に[連邦情報処理標準 \(FIPS\) 204](#) で概説されています。

ML-DSA キーは、パブリック/プライベートキーペアシステムで使用されます。プライベートキーはデータの署名に使用され、パブリックキーは署名の検証に使用されます。このシステムは、量子コンピュータの潜在的な脅威に直面しても、デジタルメッセージまたはドキュメントの信頼性、完全性、および否認の防止を保証します。

ML-DSA キー仕様でキーを作成すると、は署名と検証のために ML-DSA キーペアを持つ非対称 KMS キー AWS KMS を作成します。署名を生成するプライベートキーが暗号化 AWS KMS されていないままになることはありません。パブリックキーを使用して内部[の署名を検証する](#)か AWS KMS、外部で使用する[パブリックキーをダウンロードできます](#) AWS KMS。

AWS KMS は、非対称 KMS キーに対して次の ML-DSA キー仕様をサポートしています。

- ML_DSA_44
- ML_DSA_65
- ML_DSA_87

AWS KMS は、すべての ML-DSA キー仕様で ML_DSA_SHAKE_256 署名アルゴリズムをサポートします。

SM2 キー仕様 (中国リージョンのみ)

SM2 キー仕様は、[中国国家商業暗号局 \(OSCCA\)](#) によって公開されている GM/T シリーズの仕様で定義されている楕円曲線のキー仕様です。SM2 キー仕様は、中国リージョンでのみ利用可能です。SM2 キー仕様を使用する場合、は SM2 キーペアを使用して非対称 KMS キー AWS KMS を作成します。SM2 キーペアは、内で使用することも AWS KMS、の外部で使用するパブリックキーをダウンロードすることもできます AWS KMS。詳細については、「[the section called “SM2 キーペアによるオフライン検証 \(中国リージョンのみ\)”](#)」を参照してください。

各 KMS キーに含めることができるは 1 つだけです[Key usage](#)。SM2 KMS キーは、署名と検証、暗号化と複合、または共有シークレットの取得のいずれかに使用できます。KMS キーの作成時にキーの用途を指定する必要があります。キーの作成後に変更することはできません。

非対称 KMS キーを作成して[共有シークレットを取得する](#)場合は、SM2 キー仕様を使用します。共有シークレットの取得向けにサポートされている唯一のキーアグリーメントアルゴリズムは、[楕円曲線暗号化共同係数 Diffie-Hellman Primitive \(ECDH\)](#) です。

AWS KMS は、次の SM2 暗号化および署名アルゴリズムをサポートしています。

- SM2PKE 暗号化アルゴリズム

SM2PKE は、GM/T 0003.4-2012 で OSCCA によって定義された楕円曲線ベースの暗号化アルゴリズムです。

- SM2DSA 署名アルゴリズム

SM2DSA は、GM/T 0003.2-2012 で OSCCA によって定義された楕円曲線ベースの暗号化アルゴリズムです。SM2DSA には、SM3 ハッシュアルゴリズムでハッシュされ、渡したメッセージまたはメッセージダイジェストと組み合わせる識別 ID が必要です AWS KMS。この連結された値はハッシュ化され、署名されます AWS KMS。

AWS KMS アクセス許可

このテーブルは、AWS KMS リソースへのアクセスを制御できるように、アクセス AWS KMS 許可を理解するのに役立つように設計されています。列見出しの定義は、表の下に表示されます。

また、「サービス認可リファレンス」の「の [アクション、リソース、および条件キー AWS Key Management Service](#)」トピックでアクセス AWS KMS 許可について学ぶこともできます。ただし、このトピックには、各許可の絞り込みに使用できる条件キーがすべて一覧表示されているわけではありません。

対称暗号化 KMS キー、非対称 KMS キー、および HMAC KMS キーに対して有効な AWS KMS オペレーションの詳細については、「」を参照してください [キータイプリファレンス](#)。

Note

テーブル内のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
CancelKeyDeletion kms:CancelKeyDeletion	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService
ConnectCustomKeyStore kms:ConnectCustomKeyStore	IAM ポリシー	いいえ	*	kms:CallerAccount
CreateAlias kms:CreateAlias このオペレーションを使用するには、発信者には 2 つのリソースでの kms:CreateAlias 許可が必要です。 - エイリアス (IAM ポリシーにおける) - KMS キー (キーポリシー内) 詳細については、「 エイリアスへのアクセスの制御 」を参照してください。	IAM ポリシー (エイリアス用) キーポリシー (KMS キー用)	いいえ	エイリアス KMS キー	なし (エイリアスへのアクセスを制御する場合) KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
CreateCustomKeyStore kms:CreateCustomKeyStore	IAM ポリシー	いいえ	*	kms:CallerAccount

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
CreateGrant kms:CreateGrant	キーポリシー	はい	KMS キー	暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys 権限条件: kms:GrantConstraintType kms:GranteePrincipal kms:GrantIsForAWSResource kms:GrantOperations kms:RetiringPrincipal KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
				aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService
CreateKey kms:CreateKey	IAM ポリシー	いいえ	*	kms:BypassPolicyLookupSafetyCheck kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ViaService aws:RequestTag/tag-key (AWS グローバル条件キー) aws:ResourceTag/tag-key (AWS グローバル条件キー) aws:TagKeys (AWS グローバル条件キー)

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
Decrypt kms:Decrypt	キーポリシー	はい	KMS キー	暗号化オペレーションの条件 kms:EncryptionAlgorithm kms:RequestAlias 暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
DeleteAlias <code>kms:DeleteAlias</code> このオペレーションを使用するには、発信者には 2 つのリソースでの <code>kms:DeleteAlias</code> 許可が必要です。 - エイリアス (IAM ポリシーにおける) - KMS キー (キーポリシー内) 詳細については、「 エイリアスへのアクセスの制御 」を参照してください。	IAM ポリシー (エイリアス用)	いいえ	エイリアス	なし (エイリアスへのアクセスを制御する場合)
	キーポリシー (KMS キー用)	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService
DeleteCustomKeyStore <code>kms:DeleteCustomKeyStore</code>	IAM ポリシー	いいえ	*	kms:CallerAccount

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
DeleteImportedKeyMaterial kms:DeleteImportedKeyMaterial	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
DeriveSharedSecret kms:DeriveSharedSecret	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService 暗号化オペレーションの条件: kms:KeyAgreementAlgorithm
DescribeCustomKeyStores kms:DescribeCustomKeyStores	IAM ポリシー	いいえ	*	kms:CallerAccount

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
DescribeKey kms:DescribeKey	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件: kms:RequestAlias

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
DisableKey kms:DisableKey	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
DisableKeyRotation kms:DisableKeyRotation	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService
DisconnectCustomKeyStore kms:DisconnectCustomKeyStore	IAM ポリシー	いいえ	*	kms:CallerAccount

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
EnableKey kms:EnableKey	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
EnableKeyRotation <code>kms:EnableKeyRotation</code>	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService 自動キーローテーション条件: kms:RotationPeriodInDays

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
暗号化 kms:Encrypt	キーポリシー	はい	KMS キー	暗号化オペレーションの条件 kms:EncryptionAlgorithm kms:RequestAlias 暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GenerateDataKey <code>kms:GenerateDataKey</code>	キーポリシー	はい	KMS キー	暗号化オペレーションの条件 kms:EncryptionAlgorithm kms:RequestAlias 暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GenerateDataKeyPair <code>kms:GenerateDataKeyPair</code>	キーポリシー	はい	KMS キー 対称暗号化 KMS キーによって保護される非対称データキーペアを生成します。 。	データキーペアの条件: kms:DataKeyPairSpec 暗号化オペレーションの条件 kms:EncryptionAlgorithm kms:RequestAlias 暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
				aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GenerateDataKeyPairWithoutPlaintext <code>kms:GenerateDataKeyPairWithoutPlaintext</code>	キーポリシー	はい	KMS キー 対称暗号化 KMS キーによって保護される非対称データキーペアを生成します。 。	データキーペアの条件: kms:DataKeyPairSpec 暗号化オペレーションの条件 kms:EncryptionAlgorithm kms:RequestAlias 暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
				aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GenerateDataKeyWithoutPlaintext kms:GenerateDataKeyWithoutPlaintext	キーポリシー	はい	KMS キー	暗号化オペレーションの条件 kms:EncryptionAlgorithm kms:RequestAlias 暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GenerateMac kms:GenerateMac	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService 暗号化オペレーションの条件: kms:MacAlgorithm kms:RequestAlias
GenerateRandom kms:GenerateRandom	IAM ポリシー	該当なし	*	なし

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GetKeyPolicy kms:GetKeyPolicy	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GetKeyRotationStatus kms:GetKeyRotationStatus	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GetParametersForImport <code>kms:GetParametersForImport</code>	キーポリシー	いいえ	KMS キー	kms:WrappingAlgorithm kms:WrappingKeySpec KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
GetPublicKey kms:GetPublicKey	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件: kms:RequestAlias

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
ImportKeyMaterial kms:ImportKeyMaterial	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件: kms:ExpirationModel kms:ValidTo
ListAliases kms:ListAliases	IAM ポリシー	いいえ	*	なし

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
ListGrants kms:ListGrants	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件: kms:GrantIsForResource

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
ListKeyPolicies kms:ListKeyPolicies	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
ListKeyRotations <code>kms:ListKeyRotations</code>	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService
ListKeys <code>kms:ListKeys</code>	IAM ポリシー	いいえ	*	なし

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
ListResourceTags <code>kms:ListResourceTags</code>	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService
ListRetirableGrants <code>kms:ListRetirableGrants</code>	IAM ポリシー	指定されたプリンシパルがローカルアカウントにある必要があります。オペレーションはすべてのアカウントで権限を返します。	*	なし

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
PutKeyPolicy kms:PutKeyPolicy	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件: kms:BypassPolicyLockoutSafetyCheck

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
ReEncrypt <code>kms:ReEncryptFrom</code> <code>kms:ReEncryptTo</code> このオペレーションを使用するには、発信者に 2 つの KMS キーでのアクセス許可が必要です。 - 復号に使用される KMS キーの <code>kms:ReEncryptFrom</code> - 暗号化に使用される KMS キーの <code>kms:ReEncryptTo</code>	キーポリシー	はい	KMS キー	暗号化オペレーションの条件 kms:EncryptionAlgorithm kms:RequestAlias 暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
				その他の条件: kms:ReEncryptOnSameKey
ReplicateKey kms:ReplicateKey このオペレーションを使用するには、発信者に次のアクセス許可が必要です。 - マルチリージョンプライマリキーの kms:ReplicateKey - レプリカリージョンの IAM ポリシーの kms:CreateKey	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件: kms:ReplicaRegion

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
RetireGrant kms:RetireGrant 権限を使用停止にするアクセス許可は、主に権限によって決定されます。ポリシーだけでは、このオペレーションへのアクセスを許可することはできません。詳細については、「 グラントの使用停止と取り消し 」を参照してください。	IAM ポリシー (このアクセス許可はキーポリシーでは無効です)。	はい	KMS キー	暗号化コンテキスト条件: kms:EncryptionContext:context-key kms:EncryptionContextKeys 権限条件: kms:GrantConstraintType KMS キーオペレーションの条件: kms:CallerAccount kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
RevokeGrant kms:RevokeGrant	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件: kms:GrantIsForResource

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
RotateKeyOnDemand kms:RotateKeyOnDemand	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
ScheduleKeyDeletion kms:ScheduleKeyDeletion	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
Sign kms:Sign	キーポリシー	はい	KMS キー	署名および検証の条件: kms:MessageType kms:RequestAlias kms:SigningAlgorithm KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
TagResource kms:TagResource	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService タグ付けの条件: aws:RequestTag/tag-key (AWS グローバル条件キー) aws:TagKeys (AWS グローバル条件キー)

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
UntagResource kms:UntagResource	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService タグ付けの条件: aws:RequestTag/tag-key (AWS グローバル条件キー) aws:TagKeys (AWS グローバル条件キー)

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
UpdateAlias <code>kms:UpdateAlias</code> このオペレーションを使用するには、発信者には 3 つのリソースでの <code>kms:UpdateAlias</code> 許可が必要です。 - エイリアス - 現在関連付けられている KMS キー - 新しく関連付けられた KMS キー 詳細については、「 エイリアスへのアクセスの制御 」を参照してください。	IAM ポリシー (エイリアス用)	いいえ	エイリアス	なし (エイリアスへのアクセスを制御する場合)
	キーポリシー (KMS キー用)	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService
UpdateCustomKeyStore <code>kms:UpdateCustomKeyStore</code>	IAM ポリシー	いいえ	*	kms:CallerAccount

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
UpdateKeyDescription kms:UpdateKeyDescription	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
UpdatePrimaryRegion <code>kms:UpdatePrimaryRegion</code> 発信者がこのオペレーションを使用するには、レプリカキーとなる マルチリージョンプライマリキー と、プライマリキーとなる マルチリージョンレプリカキー の両方に対する <code>kms:UpdatePrimaryRegion</code> のアクセス許可が必要です。	キーポリシー	いいえ	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService その他の条件 kms:PrimaryRegion

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
Verify kms:Verify	キーポリシー	はい	KMS キー	署名および検証の条件: kms:MessageType kms:RequestAlias kms:SigningAlgorithm KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService

アクションおよびアクセス許可	ポリシータイプ	クロスアカウントの使用	リソース (IAM ポリシー用)	AWS KMS 条件キー
VerifyMac kms:VerifyMac	キーポリシー	はい	KMS キー	KMS キーオペレーションの条件: kms:CallerAccount kms:KeySpec kms:KeyUsage kms:KeyOrigin kms:MultiRegion kms:MultiRegionKeyType kms:ResourceAliases aws:ResourceTag/tag-key (AWS グローバル条件キー) kms:ViaService 暗号化オペレーションの条件: kms:MacAlgorithm kms:RequestAlias

列の説明

このテーブルの列には、以下の情報が表示されます:

- アクションとアクセス許可には、各 AWS KMS API オペレーションと、オペレーションを許可するアクセス許可が一覧表示されます。ポリシーステートメントの Action 要素でオペレーションを指定します。

- ポリシータイプは、アクセス許可がキーポリシーまたは IAM ポリシーで使用できるかどうかを表示します。

キーポリシーは、キーポリシーでアクセス許可を指定できることを意味します。キーポリシーに [IAM ポリシーを有効にするポリシーステートメント](#) が含まれている場合には、IAM ポリシーで許可を指定できます。

IAM ポリシーは、IAM ポリシーでのみアクセス許可を指定できることを意味します。

- クロスアカウント使用は、別の AWS アカウントで、認可されたユーザーが実行できるオペレーションを表示します。

はいの値は、別の AWS アカウントで、プリンシパルがリソースに対してオペレーションを実行できることを意味します。

いいえの値は、自分の AWS アカウントで、プリンシパルがリソースに対してのみオペレーションを実行できることを意味します。

別のアカウントのプリンシパルにクロスアカウントリソースで使用できないアクセス許可を付与した場合、そのアクセス許可は無効になります。例えば、別のアカウントでプリンシパルに、ユーザーのアカウントの KMS キーへの [kms:TagResource](#) 許可を付与した場合、KMS キーをユーザーのアカウントへタグ付けする試行は失敗します。

- リソースには、アクセス許可が適用される AWS KMS リソースが一覧表示されます。は、KMS キーとエイリアスの 2 つのリソースタイプ AWS KMS をサポートします。キーポリシーでは、Resource 要素の値は常に * であり、キーポリシーがアタッチされている KMS キーを示します。

IAM ポリシーで AWS KMS リソースを表すには、次の値を使用します。

KMS キー

リソースが KMS キーの場合は、その [キー ARN](#) を使用します。ヘルプについては、「[the section called “キー ID とキー ARN を検索する”](#)」を参照してください。

`arn:AWS_partition_name:kms:AWS_Region:AWS_account_ID:key/key_ID`

例:

`arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab`

エイリアス

リソースがエイリアスの場合は、その[エイリアス ARN](#) を使用します。ヘルプについては、「[the section called “エイリアス名とエイリアス ARN を見つける”](#)」を参照してください。

`arn:AWS_partition_name:kms:AWS_region:AWS_account_ID:alias/alias_name`

例:

`arn:aws:kms:us-west-2:111122223333:alias/ExampleAlias`

* (アスタリスク)

アクセス許可が特定のリソース (KMS キーまたはエイリアス) に適用されない場合は、アスタリスク (*) を使用します。

アクセス AWS KMS 許可の IAM ポリシーでは、Resource 要素のアスタリスクはすべての AWS KMS リソース (KMS キーとエイリアス) を示します。アクセス AWS KMS 許可が特定の KMS キーまたはエイリアスに適用されない場合は、Resource 要素でアスタリスクを使用することもできます。たとえば、`kms:CreateKey` または `アクセスkms:ListKeys` 許可を許可または拒否する場合は、Resource 要素を `*` に設定する必要があります*。

- AWS KMS 条件キーには、オペレーションへのアクセスを制御するために使用できる AWS KMS 条件キーが一覧表示されます。ポリシーの Condition 要素で条件を指定します。詳細については、「[AWS KMS 条件キー](#)」を参照してください。この列には、でサポートされている[AWS グローバル条件キー](#)も含まれていますが AWS KMS、すべての AWS サービスでサポートされているわけではありません。

AWS KMS 内部オペレーション

AWS Key Management Service (AWS KMS) は、クラウド向けにスケーリングされた [FIPS 140-3 セキュリティレベル 3 検証済みハードウェアセキュリティモジュール \(HSM\)](#) によって保護された暗号化キーとオペレーションを提供します。AWS KMS キーと機能は複数の AWS クラウドサービスで使用され、アプリケーション内のデータを保護するために使用できます。このテクニカルガイドでは、の使用 AWS 時に 内で実行される暗号化オペレーションについて詳しく説明します AWS KMS。

AWS KMS 内部は、グローバルに分散されたキー管理サービスの HSMs をスケーリングして保護するために必要です。

トピック

- [ドメインとドメインの状態](#)
- [内部通信セキュリティ](#)
- [マルチリージョンキーのレプリケーションプロセス](#)
- [耐久性の保護](#)

ドメインとドメインの状態

内の信頼できる内部 AWS KMS エンティティの協調的なコレクション AWS リージョン は、ドメインと呼ばれます。ドメインには、信頼済みエンティティのセット、ルール of セット、およびドメインキーと呼ばれる秘密キーのセットが含まれます。ドメインキーは、ドメインのメンバーである HSM 間で共有されます。ドメイン状態は次のフィールドで構成されます。

名前

このドメインを識別するドメイン名。

メンバー

ドメインのメンバーである HSM のリスト。パブリック署名キーとパブリック共有キーを含む。

演算子

このサービスのオペレーターを表すエンティティ、パブリック署名キー、ロール (AWS KMS オペレーターまたはサービスホスト) のリスト。

ルール

HSM でコマンドを実行するために満たす必要がある各コマンドの定足数ルールのリスト。

ドメインキー

ドメイン内で現在使用されているドメインキー (対称キー) のリスト。

完全なドメイン状態は、HSM 上でのみ取得できます。ドメイン状態は、エクスポートされたドメイントークンとして HSM ドメインメンバー間で同期されます。

ドメインキー

ドメイン内のすべての HSM は、ドメインキーのセット $\{DK_i\}$ を共有します。これらのキーは、ドメイン状態のエクスポートルーチンで共有されます。エクスポートされたドメイン状態は、ドメインのメンバーになっている任意の HSM にインポートできます。

ドメインキーのセット {DK_r} には、常に 1 つのアクティブなドメインキーと、いくつかの非アクティブなドメインキーが含まれます。ドメインキーは、[ガキー管理の推奨事項 - パート 1](#) AWS に準拠するように毎日ローテーションされます。ドメインキーのローテーション中に、送信ドメインキーの下で暗号化された既存の KMS キーはすべて、新しいアクティブなドメインキーの下で再暗号化されます。アクティブなドメインキーが、新しい EKT の暗号化には使用されます。期限切れのドメインキーは、最近ローテーションされたドメインキーの数と同等の日数の間、以前に暗号化された EKT を復号化するためにのみ使用できます。

エクスポートされたドメイントークン

ドメイン参加者間で定期的に状態を同期させる必要があります。これは、ドメインに変更が加えられるたびにドメインの状態をエクスポートすることによって実現されます。ドメインの状態は、エクスポートされたドメイントークンとしてエクスポートされます。

名前

このドメインを識別するドメイン名。

メンバー

ドメインのメンバーである HSM のリスト。署名パブリックキーおよび共有パブリックキーを含む。

演算子

エンティティ、パブリック署名キー、およびこのサービスのオペレーターを表すロールのリスト。

ルール

HSM ドメインメンバーでコマンドを実行するために満たす必要がある各コマンドの定足数ルールのリスト。

暗号化されたドメインキー

エンベロープで暗号化されたドメインキー。ドメインキーは、上記の各メンバーの署名メンバーによって暗号化され、パブリック共有キーにエンベロープされます。

署名

HSM によって生成されるドメイン状態の署名。ドメイン状態をエクスポートしたドメインのメンバーである必要があります。

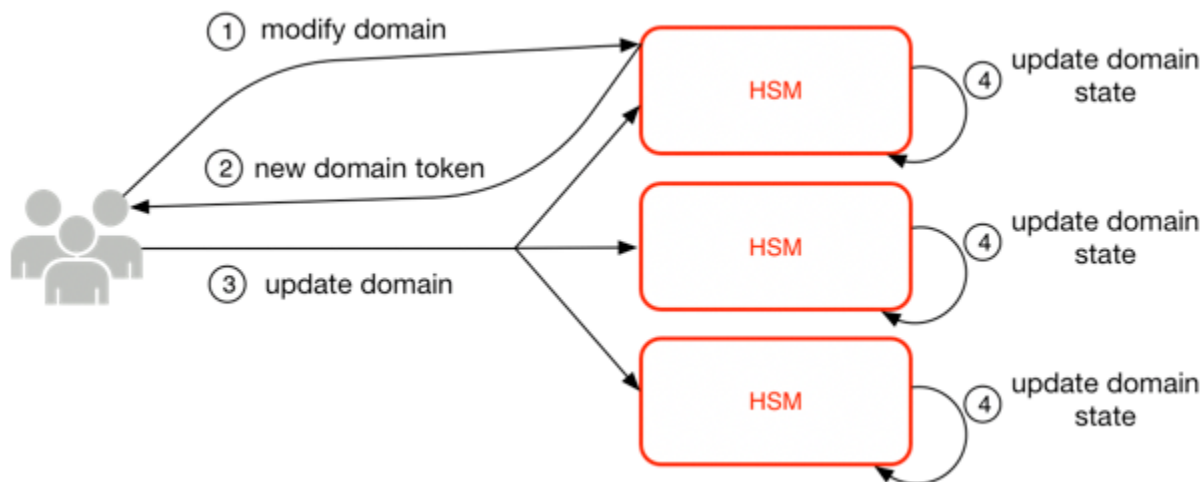
エクスポートされたドメイントークンは、ドメイン内で動作するエンティティの信頼の基盤を形成します。

ドメイン状態の管理

ドメインの状態は、定足数認証コマンドを使用して管理されます。これらの変更には、ドメイン内の信頼済み参加者のリストの変更、HSM コマンドを実行するための定足数ルールの変更、ドメインキーの定期的なローテーションが含まれます。これらのコマンドは、次の図に示すように、認証されたセッションオペレーションではなく、コマンド単位で認証されます。

初期化され動作状態にある HSM は、自己生成の非対称 ID キー、署名キーペア、およびキー確立キーペアのセットを含んでいます。手動プロセスにより、AWS KMS オペレーターはリージョンの最初の HSM に作成される初期ドメインを確立できます。この初期ドメインは、このトピックで以前に定義した完全なドメイン状態で構成されます。ドメイン内の定義済みの HSM メンバーごとに、join コマンドを使用してインストールされます。

HSM が初期ドメインに参加すると、そのドメインで定義されているルールにバインドされます。これらのルールは、お客様の暗号化キーを使用するコマンドや、ホストまたはドメインの状態を変更するコマンドを管理します。暗号化キーを使用する認証済みセッション API オペレーションは、以前に定義されています。



上の図は、ドメインの状態がどのように変更されるかを示しています。このプロセスは次の 4 つのステップで構成されています。

1. ドメインを変更するために、定足数ベースのコマンドが HSM に送信します。
2. 新しいドメイン状態が生成され、新しいエクスポートされたドメイントークンとしてエクスポートされます。HSM 上の状態は変更されません。つまり、HSM 上で変更が行われません。

3. 2 番目のコマンドが新しくエクスポートされたドメイントークンの各 HSM に送信され、ドメインの状態を新しいドメイントークンで更新します。
4. エクスポートされた新しいドメイントークンにリストされている HSM は、コマンドとドメイントークンを認証できます。また、ドメイン内のすべての HSM のドメイン状態を更新するために、ドメインキーを解凍することもできます。

HSM は互いに直接通信しません。代わりに、定足数のオペレーターがドメインの状態の変更を要求し、結果として新しいエクスポートされたドメイントークンを生成します。ドメインのサービスホストメンバーが、ドメイン内のすべての HSM に新しいドメイン状態を配布するために使用されます。

ドメインの脱退と参加は、HSM 管理機能によって行われます。ドメイン状態の変更は、ドメイン管理機能によって行われます。

ドメインからの脱退

HSM はドメインから脱退し、すべての残留物とそのドメインのキーがメモリから削除されます。

ドメインへの参加

HSM は新しいドメインに参加するか、現在のドメイン状態を新しいドメイン状態に更新します。既存のドメインは、このメッセージを認証するためのルールの初期セットのソースとして使用されます。

ドメインの作成

HSM 上に新しいドメインを作成します。ドメインのメンバー HSM に配布できる最初のドメイントークンを返します。

オペレーターの変更

許可されたオペレーターとドメイン内のオペレーターのロールのリストにオペレーターを追加または削除します。

メンバーの変更

ドメイン内の許可された HSM のリストに HSM を追加または削除します。

ルールの変更

HSM でコマンドを実行するために必要な定足数ルールのセットを変更します。

ドメインキーのローテーション

新しいドメインキーが作成され、アクティブドメインキーとしてマークされます。これにより、既存のアクティブキーが非アクティブ化されたキーになり、最も古い非アクティブ化されたキーがドメイン状態から削除されます。

内部通信セキュリティ

サービスホストまたは AWS KMS オペレータと HSMs 間のコマンドは、に示されている [認証済みセッション](#)2 つのメカニズム、つまりクォーラム署名付きリクエストメソッドと HSM サービスホストプロトコルを使用した認証済みセッションによって保護されます。

定足数署名付きコマンドは、HSM が提供する重要なセキュリティ保護を 1 人のオペレーターが改変できないように設計されています。認証されたセッションで実行されるコマンドは、許可されたサービスオペレーターだけが KMS キーに関連する操作を実行できるようにするのに役立ちます。顧客宛ての機密情報はすべて、AWS インフラストラクチャ全体で保護されます。

キー確立

内部通信を保護するために、は 2 つの異なるキー確立方法 AWS KMS を使用します。1 つ目は、[Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography](#) の「C(1, 2, ECC DH)」として定義されています。このスキームには、静的な署名キーを持つイニシエータがあります。イニシエータは、静的 ECDH 共有キーを持つ受信者を対象として、一時的な楕円曲線 Diffie-Hellman (ECDH) キーを生成し、署名します。このメソッドは、ECDH を使用した 1 つの一時的キーと 2 つの静的キーを使用します。これは、ラベル C(1, 2, ECC DH) の導出です。この方法は、ワンパス ECDH と呼ばれることもあります。

2 つ目のキー確立方法は、[\(2, 2, ECC, DH\)](#) です。この方式では、両当事者は静的な署名キーを持ち、一時的な ECDH キーを生成、署名、交換します。この方法では、ECDH をそれぞれ使用した 2 つの静的キーと 2 つの一時的キーを使用します。これは、ラベル C(2, 2, ECC, DH) の導出です。この方法は、ECDH Ephemeral または ECDHE と呼ばれることもあります。すべての ECDH キーは、曲線 secp384r1 (NIST-P384) 上で生成されます。

HSM セキュリティ境界

の内部セキュリティ境界 AWS KMS は HSM です。HSM には独自のインターフェイスがあり、動作状態では他のアクティブな物理インターフェイスはありません。動作中の HSM は、ドメイン内でのロールを確立するために必要な暗号化キーを使用して初期化中にプロビジョニングされます。HSM

の機密暗号化マテリアルは、揮発性メモリにのみ保存され、意図的であるかないかにかかわらず、シャットダウンやリセットなど、動作状態でなくなると消去されます。

HSM API オペレーションは、個々のコマンドによって、またはサービスホストによって確立された相互に認証された機密セッションによって認証されます。

定足数署名付きコマンド

定足数署名付きコマンドは、オペレーターによって HSM に対して発行されます。このセクションでは、定足数ベースのコマンドを作成、署名、および認証する方法について説明します。これらのルールはかなり簡単です。例えば、コマンド Foo には、ロール Bar からの 2 人のメンバーが必要といったものです。定足数ベースのコマンドの作成と検証には、3 つのステップがあります。最初のステップは元になるコマンドの作成で、2 番目のステップは署名する追加のオペレーターへの送信、3 番目のステップは検証と実行です。

概念を理解するため、次のように仮定します。オペレーターのパブリックキーとロールの信頼できるセット $\{QOS_s\}$ があり、一連の定足数ルール $QR = \{Command_i, Rule_{\{i, t\}}\}$ があるとします。ここで、各ルールはロールと最小数 $N \{Role_t, N_t\}$ のセットにより構成されます。コマンドが定足数ルールを満たすためには、 $\{QOS_s\}$ にリストされている一連のオペレーターが、そのコマンド用にリストされているルールのいずれかを満たすように、コマンドデータセットに署名する必要があります。前述のように、定足数ルールとオペレーターのセットは、ドメイン状態とエクスポートされたドメイントークンに保存されます。

実際には、最初の署名者はコマンド $Sig_1 = \text{Sign}(dOp_1, \text{Command})$ に署名します。2 番目のオペレーターもコマンド $Sig_2 = \text{Sign}(dOp_2, \text{Command})$ に署名します。2 者に署名されたメッセージは、実行のために HSM に送信されます。HSM は、次の内容を実行します。

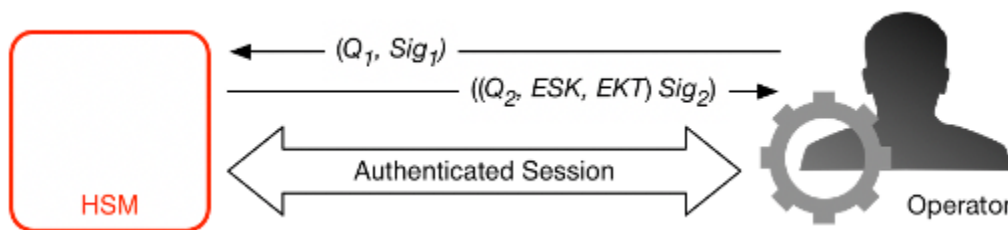
1. 各署名ごとに、ドメイン状態から署名者のパブリックキーが抽出され、コマンドの署名が検証されます。
2. 署名者のセットがコマンドのルールを満たしていることを確認します。

認証済みセッション

キーオペレーションは、外部向け AWS KMS ホストと HSMs の間で実行されます。これらのコマンドは、暗号キーの作成と使用、および安全な乱数の生成に関係します。コマンドは、サービスホストと HSM の間のセッション認証チャンネル上で実行されます。これらのセッションには、真正性が必要なだけでなく、機密性も必要です。これらのセッションで実行されるコマンドは、発行者に向けてクリアテキストのデータキーや復号化されたメッセージを返送することがあります。中間者 (MITM) 攻撃によってこれらのセッションが崩壊しないようにするため、セッションは認証されます。

このプロトコルは、HSM とサービスホスト間で相互に認証された ECDHE キー共有を実行します。交換はサービスホストが開始し、HSM が完了させます。HSM は、ネゴシエートされたキーによって暗号化されたセッションキー (SK) と、セッションキーを含むエクスポートされたキートークンも返します。エクスポートされたキートークンには有効期間が含まれています。有効期間が終了すると、サービスホストはセッションキーを再ネゴシエートする必要があります。

サービスホストはドメインのメンバーであり、ID 署名キーペア ($dHOS_i$, $QHOS_i$) と、HSM の ID パブリックキーの信頼できるコピーを持っています。サービスホストは、ID 署名キーのセットを使用して、サービスホストとドメイン内の任意の HSM との間で使用するセッションキーを安全にネゴシエートします。エクスポートされたキートークンには有効期間が関連付けられています。有効期間が終了すると、新しいキーをネゴシエートする必要があります。



このプロセスは、サービスホストが、自身とドメインの HSM メンバー間で機密通信フローを送受信するにはセッションキーが必要であることを認識することから始まります。

1. サービスホストが ECDH 一時的キーペア (d_1 , Q_1) を生成し、これに ID キー $Sig_1 = \text{Sign}(dOS, Q_1)$ で署名します。
2. HSM は、受信したパブリックキーの署名を現在のドメイントークンを使用して検証し、ECDH 一時的キーペア (d_2 , Q_2) を作成します。次に、離散対数暗号を使用したペアワイズキー確立スキームの推奨事項に従って ECDH-key-exchange を完了し、ネゴシエートされた 256 ビット AES-GCM キーを作成します。 <https://csrc.nist.gov/pubs/sp/800/56/a/r3/final> HSM は、新しい 256 ビット AES-GCM セッションキーを生成します。ネゴシエートされたキーを使用してセッションキーを暗号化し、暗号化されたセッションキー (ESK) を形成します。また、ドメインキーの下にあるセッションキーをエクスポートされたキートークン EKT として暗号化します。最後に、ID キーペア $Sig_2 = \text{Sign}(dHSM, (Q_2, ESK, EKT))$ で戻り値に署名します。
3. サービスホストは、現在のドメイントークンを使用して、受信したキーの署名を検証します。次に、サービスホストは、[離散対数暗号を使用したペアワイズキー確立スキームの推奨事項](#)に従って ECDH キー交換を完了します。次に ESK を復号化し、セッションキー SK を取得します。

EKT の有効期間中は、サービスホストはネゴシエートされたセッションキー SK を使用して、エンベロープで暗号化されたコマンドを HSM に送信できます。この認証されたセッションでサービス

ホストが開始するコマンドにはすべて、EKT が含まれています。HSM は、ネゴシエートされた同じセッションキー SK を使用して応答します。

マルチリージョンキーのレプリケーションプロセス

AWS KMS は、クロスリージョンレプリケーションメカニズムを使用して、KMS キーのキーマテリアルを の HSM から別の の HSM AWS リージョン にコピーします AWS リージョン。このメカニズムを機能させるには、レプリケートされる KMS キーがマルチリージョンキーである必要があります。KMS キーをあるリージョンから別のリージョンにレプリケートする場合、各リージョン内の HSM はそれぞれ分離されたネットワーク内にあるため、直接通信できません。代わりに、クロスリージョンレプリケーション中に交換されるメッセージは、プロキシサービスによって配信されます。

クロスリージョンレプリケーションでは、AWS KMS HSM によって生成されたすべてのメッセージは、レプリケーション署名キーを使用して暗号で署名されます。レプリケーション署名キー (RSK) は、NIST P-384 曲線上の ECDSA キーです。各リージョンは少なくとも 1 つの RSK を所有し、各 RSK のパブリックコンポーネントは同じ AWS パーティション内の他のすべてのリージョンと共有されます。

リージョン A からリージョン B にキーマテリアルをコピーするクロスリージョンレプリケーションプロセスは、次のように機能します。

1. リージョン B の HSM は、NIST P-384 曲線上に一時的な ECDH キーを生成します。Replication Agreement Key (レプリケーションアグリーメントキー) B (RAKB) です。RAKB のパブリックコンポーネントは、プロキシサービスによってリージョン A の HSM に送信されます。
2. リージョン A の HSM は RAKB のパブリックコンポーネントを受け取り、NIST P-384 曲線上に別の一時的な ECDH キーを生成します。Replication Agreement Key (レプリケーションアグリーメントキー) A (RAKA) です。HSM は、RAKA と RAKB のパブリックコンポーネント上で ECDH キー確立スキームを実行し、出力から対称キーを導出します。Replication Wrapping Key (レプリケーションラッピングキー) (RWK) です。RWK は、レプリケートされるマルチリージョン KMS キーのキーマテリアルを暗号化するために使用されます。
3. RAKA のパブリックコンポーネントと RWK で暗号化されたキーマテリアルは、プロキシサービスを通じてリージョン B の HSM に送信されます。
4. リージョン B の HSM は、RAKA のパブリックコンポーネントと RWK を使用して暗号化されたキーマテリアルを受け取ります。HSM は、RAKB と RAKA のパブリックコンポーネント上で ECDH キー確立スキームを実行し RWK で派生します。
5. リージョン B の HSM は、RWK を使用してリージョン A のキーマテリアルを復号化します。

耐久性の保護

オフライン HSM の使用、エクスポートされたドメイントークンの複数の不揮発性ストレージ、および暗号化された KMS キーの冗長ストレージにより、サービスによって生成されたキーに対するサービスの耐久性が強化されます。オフライン HSM は、既存のドメインのメンバーです。オンラインではなく、通常のドメイン操作に参加している場合を除き、オフライン HSM は既存の HSM メンバーと同じドメイン状態で表示されます。

耐久性設計は、オンライン HSMs またはプライマリストレージシステム内に保存されている KMS キーのセットのいずれかが広範囲に失われ AWS の場合に、リージョン内のすべての KMS キーを保護することを目的としています。インポートされたキーマテリアル AWS KMS keys を持つ は、他の KMS キーに許容される耐久性保護には含まれません。でリージョン全体の障害が発生した場合 AWS KMS、インポートされたキーマテリアルを KMS キーに再インポートする必要がある場合があります。

オフラインの HSM と、それらにアクセスするための認証情報は、複数の独立した地理的場所にある監視された安全室内の金庫に保存されます。各金庫には、これらの資料を取得するために、の 2 つの独立したチームから少なくとも 1 人の AWS セキュリティ責任者と 1 AWS 人の AWS KMS オペレーターが必要です。これらのマテリアルの使用は、クォーラムの AWS KMS 演算子の存在を要求する内部ポリシーによって管理されます。

ドキュメント履歴

このトピックでは、AWS Key Management Service デベロッパーガイドの重要な更新を説明しています。

トピック

- [最新の更新](#)
- [以前の更新](#)

最新の更新

以下の表は、このドキュメントの 2018 年 1 月以降の大きな変更点をまとめたものです。ここに表示されている主要な変更に加えて、その内容の説明と例を向上し、ユーザーから寄せられるフィードバックにも応える目的で、このドキュメントは頻繁に更新されます。重要な変更についての通知を受け取るには、RSS フィードをサブスクライブします。

この表のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

変更	説明	日付
デュアルスタックのサポート	AWS KMS はデュアルスタックをサポートしています。	2025 年 6 月 18 日
機能更新	Module-Lattice Digital Signature Algorithm (ML-DSA) ポスト量子暗号化署名のサポートを追加しました。	2025 年 6 月 13 日
インポートされたキーローテーション	インポートされたキーマテリアル (EXTERNALオリジン) を使用して、対称暗号化 KMS キーのオンデマンドローテーションを実行できます。	2025 年 6 月 5 日
機能更新	中国リージョンでのマルチリージョン KMS キーのサポートが追加されました。	2024 年 11 月 21 日

[AWS マネージドポリシーの更新](#)

ポリシーバージョン v2 の マネージドポリシーにステートメント ID (Sid) を追加して、AWSKeyManagementServiceMultiRegionKeysServiceRolePolicy サービスにリンクされたロールを更新しました。

2024 年 11 月 21 日

[クォータの変更](#)

[ImportKeyMaterial](#) および [DeleteImportedKeyMaterial](#) リクエストのデフォルトのリクエスト料金が引き上げられました。

2024 年 7 月 23 日

[クォータの変更](#)

対称暗号化 KMS キー、RSA KMS キー、および ECC と SM2 の KMS キー向けのデフォルトの暗号化オペレーションリクエスト料金が引き上げられました。

2024 年 7 月 8 日

[新機能](#)

NIST 推奨楕円曲線 (ECC) KMS キーと SM2 (中国リージョンのみ) KMS キーに新しい KeyUsage タイプの KEY_AGREEMENT を追加し、[共有シークレットの取得](#)のサポートを追加しました。

2024 年 6 月 13 日

キーローテーションの更新	自動キーローテーション向けカスタムローテーション期間、オンデマンドキーローテーション、およびキーマテリアルローテーションの可視化のサポートを追加しました。	2024 年 4 月 12 日
マネージドポリシーの更新	AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy クラスターを含む VPC の変更を AWS KMS がモニタリング AWS CloudHSM できる新しいアクセス許可を に追加し、が障害発生時に明確なエラーメッセージを提供 AWS KMS できるようにしました。	2023 年 11 月 10 日
機能更新	DryRun API パラメータのサポートを追加しました。	2023 年 7 月 5 日
機能更新	カスタムキーストアを除くすべてのタイプの AWS KMS キーのキーマテリアルのインポートのサポートが追加されました。	2023 年 6 月 5 日
機能更新	Nitro Enclaves AWS KMS APIs の更新	2023 年 3 月 10 日

機能更新

RSAES_PKCS1_V1_5 ラップアルゴリズムは廃止されました。AWS KMS は、米国国立標準技術研究所 (NIST) の[暗号化キー管理ガイド](#)に従って、2023 年 10 月 1 RSAES_PKCS1_V1_5 日までにすべてのサポートを終了します。別のラップアルゴリズムをすぐに使い始めることをお勧めします。

機能更新

外部キーストアのサポートが追加されました。これは、外部で暗号化キーを使用してリソースを AWS 保護できる機能です AWS。

クォータの変更

各アカウントとリージョンで AWS KMS keys リソースクォータを 100,000 KMS キーに引き上げました。

機能更新

で HMAC KMS キーのサポートを追加 AWS リージョン

新しいトピック

「AWS KMS デベロッパーガイド」の「セキュリティ」の章に「[の耐障害性 AWS Key Management Service](#)」トピックを追加しました。

新機能

HMAC コードを生成して検証する AWS KMS キーと API オペレーションのサポートが追加されました。

ドキュメントの変更

カスタマーマスターキー (CMK) という条件が AWS KMS key および KMS キーに置き換えられています。

2021 年 8 月 30 日

新機能

マルチリージョンキーのサポートの追加、これは、同じキー ID とキーマテリアルを持つ異なるリージョンで、相互運用可能な KMS キーのセットです。マルチリージョンキーを使用して、あるリージョンのデータを暗号化し、別のリージョンのデータを復号できます。

2021 年 6 月 8 日

新機能

属性ベースのアクセスコントロール (ABAC) のサポートを追加しました。タグとエイリアスを使用して、へのアクセスを制御できます AWS KMS keys。

2020 年 12 月 17 日

新機能

VPC エンドポイントポリシーのサポートが追加されました。

2020 年 7 月 9 日

新しいコンテンツ

のセキュリティプロパティについて説明します AWS KMS。

2020 年 6 月 18 日

新機能

非対称データキー AWS KMS keys と非対称データキーのサポートが追加されました。

2019 年 11 月 25 日

更新された機能

のキーポリシーは、AWS KMS コンソール AWS マネージドキー で表示できます。この機能は、以前はカスターマネージドキーに限定されていました。

2019 年 11 月 15 日

新機能

AWS KMSへの呼び出しに、TLS で[ハイブリッドポスト量子キー交換](#)アルゴリズムを使用する方法について説明します。

2019 年 11 月 4 日

クォータの変更

KMS キーを管理する一部の API のリソースクォータを増やしました。

2019 年 9 月 18 日

クォータの変更

KMS キーごとの KMS キー、エイリアス、グラントのリソースクォータを変更しました。

2019 年 3 月 27 日

クォータの変更

カスタムキーストアで AWS KMS keys を使用する暗号化オペレーションに対して、1 秒あたりの共有リクエストクォータを変更しました。

2019 年 3 月 7 日

新機能

AWS KMS [カスタムキーストア](#)を作成および管理する方法を説明します。各キーストアは、ユーザーが所有および管理する AWS CloudHSM クラスタによってバックアップされます。

2018 年 11 月 26 日

新しいコンソール	IAM AWS KMS コンソールから独立した新しいコンソールを使用する方法について説明します。元のコンソールとその使用方法是、新しいコンソールに精通できるように、短期間ですが引き続き参照できます。	2018 年 11 月 7 日
クォータの変更	を使用するための共有 リクエストクォータ を変更しました AWS KMS keys。	2018 年 8 月 21 日
新しいコンテンツ	がキー AWS Secrets Manager を使用してシークレットのシークレット値を暗号化する方法について説明します AWS KMS 。	2018 年 7 月 13 日
新しいコンテンツ	DynamoDB が AWS KMS AWS KMS keys を使用してサーバー側の暗号化オプションをサポートする方法について説明します。	2018 年 5 月 23 日
新機能	インターネット経由で接続するのではなく AWS KMS、 VPC 内のプライベートエンドポイントを使用して に直接接続する方法について説明します。	2018 年 1 月 22 日

以前の更新

次の表は、2018 年以前の AWS Key Management Service デベロッパーガイドの重要な変更点を示しています。

この表のすべてのデータを表示するには、水平または垂直にスクロールする必要があります。

変更	説明	日付
新しいコンテンツ	のタグ AWS KMS に関するドキュメントを追加しました。	2017 年 2 月 15 日
新しいコンテンツ	モニタリング AWS KMS keys および Amazon CloudWatch で KMS キーをモニタリングする に関するドキュメントを追加しました。	2016 年 8 月 31 日
新しいコンテンツ	インポートされたキーマテリアル に関するドキュメントを追加しました。	2016 年 8 月 11 日
新しいコンテンツ	IAM ポリシー 、 アクセス許可に関するリファレンス 、および 条件キー のドキュメントを追加しました。	2016 年 7 月 5 日
更新	ドキュメントの「 KMS キーのアクセスとアクセス許可 」章の一部を更新しました。	2016 年 7 月 5 日
更新	新しいデフォルトクォータを反映するように クォータ ページを更新しました。	2016 年 5 月 31 日
更新	新しいデフォルトのクォータを反映するように クォータ ページを更新し、わかりやすさと正確さを向上させるために グラントトークン ドキュメントを更新しました。	2016 年 4 月 11 日
新しいコンテンツ	複数の IAM プリンシパルが KMS キーにアクセスできるよ	2016 年 2 月 17 日

変更	説明	日付
	うにする および IP アドレス条件の使用 に関するドキュメントを追加しました。	
更新	明確さと正確性を向上させるため、「 のキーポリシー AWS KMS 」および「 キーポリシーを変更する 」ページを更新しました。	2016 年 2 月 17 日
更新	「KMS キーの管理」のトピックページを更新してより分かりやすくしました。	2016 年 1 月 5 日
新しいコンテンツ	CloudTrail に関するドキュメントを追加しました。	2015 年 11 月 18 日
新しいコンテンツ	「 キーポリシーを変更する 」の手順を追加しました。	2015 年 11 月 18 日
更新	Amazon Relational Database Service の使用方法に関するドキュメントを更新しました AWS KMS。	2015 年 11 月 18 日
新しいコンテンツ	Amazon WorkSpaces に関するドキュメントを追加しました。	2015 年 11 月 6 日
更新	内容をより明確にするために [のキーポリシー AWS KMS] ページが更新されました。	2015 年 10 月 22 日

変更	説明	日付
新しいコンテンツ	を削除する AWS KMS key に関するドキュメント (アラームの作成 、 KMS キーの過去の使用状況を確認する の関連ドキュメントなど) を追加しました。	2015 年 10 月 15 日
新しいコンテンツ	へのアクセスの確認 AWS KMS keys に関するドキュメントを追加しました。	2015 年 10 月 15 日
新しいコンテンツ	キーの AWS KMS キーステータス に関するドキュメントを追加しました。	2015 年 10 月 15 日
新しいコンテンツ	Amazon Simple Email Service に関するドキュメントを追加しました。	2015 年 10 月 1 日
更新	新しいリクエストクォータの説明がある クォータ ページを更新しました。	2015 年 8 月 31 日
新しいコンテンツ	の使用料金に関する情報を追加しました AWS KMS。 「 AWS KMS 料金表 」を参照してください。	2015 年 8 月 14 日
新しいコンテンツ	にリクエストクォータを追加しました AWS KMS クォータ 。	2015 年 6 月 11 日
新しいコンテンツ	UpdateAlias オペレーションの使用方法を示す新しい Java コードサンプルを追加しました。	2015 年 6 月 1 日

変更	説明	日付
更新	AWS Key Management Service リージョンの一覧 を『AWS 全般のリファレンス』に移動しました。	2015 年 5 月 29 日
新しいコンテンツ	Amazon EMR が使用する 方法 AWS KMS に関するドキュメントを追加しました。	2015 年 1 月 28 日
新しいコンテンツ	Amazon WorkMail に関するドキュメントを追加しました。	2015 年 1 月 28 日
新しいコンテンツ	Amazon Relational Database Service の使用方法に関するドキュメントを追加しました AWS KMS。	2015 年 1 月 6 日
新しいコンテンツ	Amazon Elastic Transcoder に関するドキュメントを追加しました。	2014 年 11 月 24 日
新規ガイド	AWS Key Management Service デベロッパーガイドを導入しました。	2014 年 11 月 12 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。