aws

ユーザーガイド

AWS CloudHSM



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS CloudHSM: ユーザーガイド

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

とは AWS CloudHSM 1
ユースケース 2
仕組み 4
クラスター 5
のユーザー AWS CloudHSM 6
のキー AWS CloudHSM 6
クライアント SDK 7
バックアップ 8
でサポートされているリージョン AWS CloudHSM 9
の料金 AWS CloudHSM 9
開始方法 10
IAM 管理者の作成 10
IAM ユーザーおよび管理者グループの作成 11
「VPC を作成する」 11
クラスターを作成する 12
クラスターセキュリティグループの確認 16
EC2 クライアントの起動 17
EC2 インスタンスのセキュリティグループを設定する 20
ステップ 1. デフォルトのセキュリティグループの変更 21
ステップ 2. Amazon EC2 インスタンスを AWS CloudHSM クラスターに接続する 21
HSM を作成する 22
HSM のアイデンティティの確認 (オプション) 24
ステップ 1. HSM からの証明書の取得 26
ステップ 2. ルート証明書の取得 29
ステップ 3. 証明書チェーンの確認 30
ステップ 4. パブリックキーの抽出と比較 31
クラスターの初期化 32
概要 32
ステップ 1. クラスター CSR の取得 33
ステップ 2. ルート CA のプライベートキーを作成する 35
ステップ 3. CSR の署名 36
ステップ 4. クラスターの初期化 46
CloudHSM CLI をインストールする 48
クラスターのアクティブ化 53

mTLS の設定 (推奨) 56
ステップ 1. トラストアンカーを作成して HSM に登録する 56
ステップ 2. の mTLS を有効にする AWS CloudHSM 60
ステップ 3. の mTLS 適用を設定する AWS CloudHSM 65
でキーを作成して使用する AWS CloudHSM 67
ベストプラクティス 69
クラスターの管理 69
クラスターをスケールしてピークトラフィックを処理する 69
高可用性対応のクラスターをアーキテクチャー 69
新しく生成されたキーの耐久性を確保するために、HSM を少なくとも 3 つ用意してください 70
クラスターへの安全なアクセス 70
ニーズに合わせてスケールすることでコストを削減 70
ユーザー管理 71
HSM ユーザーの認証情報の保護 71
ロックアウトを防ぐため、少なくとも 2 人の管理者を配置します 71
すべてのユーザー管理オペレーションでクォーラムを有効にします 71
各自の権限を制限した複数の Crypto User を作成する 72
キー管理 72
適切なキーのタイプを選択する 72
キーのストレージ制限を管理する 72
キーラッピングの管理と保護 73
アプリケーション統合 74
クライアント SDK のブートストラップ 74
認証してオペレーションを実行 74
アプリケーションのキーを効果的に管理する 75
マルチスレッドを使用してください 76
スロットリングエラーの処理 76
クラスターオペレーションのリトライを統合してください 77
ディザスタリカバリ戦略の実装 77
アプリケーションのデプロイをずらす 78
モニタリング 78
クライアントログのモニタリング 78
監査ログのモニタリング 78
モニタリング AWS CloudTrail 79
Amazon CloudWatch メトリクスのモニタリング 79

クラスター 80
クラスターアーキテクチャ 80
クラスターの同期 82
クラスターの高可用性とロードバランシング 83
クラスターモード 84
HSM タイプ 85
クラスターへの接続 87
各 EC2 インスタンス上に発行証明書を配置する 87
発行証明書の場所を指定する 87
クライアント SDK をブートストラップする 90
HSM のスケーリング 93
HSM の追加 94
HSM の削除 96
クラスターの削除 96
バックアップからクラスターを作成する 98
バックアップからのクラスターの作成 (コンソール) 98
バックアップからのクラスターの作成 (AWS CLI) 99
バックアップからクラスターを作成する (AWS CloudHSM API) 100
HSM クラスタータイプの移行 100
hsm1.medium から hsm2m.medium への移行 101
HSM ユーザー 110
CloudHSM CLI によるユーザー管理 110
前提条件 111
ユーザータイプ 112
アクセス許可テーブル 114
Admin を作成 115
CU の作成 116
すべてのユーザーを一覧表示する 117
パスワードの変更 118
ユーザーの削除 120
ユーザー MFA の管理 121
クォーラム認証の管理 (M/N) 136
CMU によるユーザー管理 160
前提条件 161
ユーザータイプ 165
アクセス許可テーブル 167

ユーザーを作成する 168
すべてのユーザーを一覧表示する 171
パスワードの変更 172
ユーザーの削除 175
ユーザー 2FA の管理 177
CMU を使用してクォーラム認証を管理する 186
キー 206
キーの同期と耐久性 206
概念 207
キーの同期について 208
クライアントキーの耐久性設定を変更する 209
クローンされた クラスター間でキーを同期する 215
AES キーラップ 215
サポートされているアルゴリズム 215
での AES キーラップの使用 AWS CloudHSM 217
信頼できるキー 218
信頼できるキーとは 219
信頼できるキーの属性 219
信頼できるキーを使用してデータキーをラップする方法 220
データキーを信頼できるキーでアンラップする方法 223
CloudHSM CLI によるキー管理 223
キーを生成する 224
キーの削除 231
共有と共有解除キー 234
キーでフィルタリング 241
キーを信頼できるものとしてマークする 249
クォーラム認証の管理 (M/N) 250
KMU によるキー管理 270
キーを生成する 271
キーのインポート 273
キーのエクスポート 275
キーの削除 277
共有と共有解除キー 278
キーを信頼できるものとしてマークする 279
クラスターのバックアップ 280
バックアップの使用 280

有効期限切れのキー、または非アクティブなユーザーの削除 281
ディザスタリカバリの検討 281
バックアップの削除 281
バックアップの復元 283
バックアップ保持の設定 284
マネージドバックアップの保持 285
リージョン間のバックアップのコピー 288
バックアップを異なるリージョン (コンソール) にコピーする 289
バックアップを異なるリージョン (AWS CLI) にコピーする 289
異なるリージョンにバックアップをコピーする (AWS CloudHSM API) 290
共有バックアップの使用 290
バックアップを共有するための前提条件 291
バックアップの共有 291
共有バックアップの共有解除 295
共有バックアップの特定 295
共有バックアップのアクセス許可 296
請求と使用量測定 296
クローンクラスター 297
関連トピック 297
リソースにタグを付ける 298
タグを追加または更新する 298
タグの一覧表示 299
タグを削除する 300
コマンドラインツール 302
設定ツール 303
クライアント SDK 5 設定ツール 304
クライアント SDK 3 設定ツール 337
CloudHSM CLI 346
サポートされているプラットフォーム 347
開始方法 349
コマンドモード 357
キー属性 358
詳細設定 366
リファレンス 373
AWS CloudHSM 管理ユーティリティ 658
サポートされているプラットフォーム 659

開始方法 ... 659
クライアント (Linux) のインストール ... 664
クライアントのインストール (Windows) ... 667
リファレンス ... 668
キー管理ユーティリティ ... 728
開始方法 ... 729
クライアント (Linux) のインストール ... 733
クライアントのインストール (Windows) ... 736
リファレンス ... 737
クライアント SDK ... 863
バージョン確認方法 ... 864
コンポーネントのサポートを比較する ... 866
PKCS #11 ライブラリ ... 866
CloudHSM 管理ユーティリティ (CMU) ... 867
キー管理ユーティリティ (KMU) ... 867
JCE プロバイダー ... 867
OpenSSL Dynamic Engine ... 867
キーストレージプロバイダー (KSP) ... 868
最新の SDK への移行 ... 868
PKCS #11 ライブラリを移行する ... 869
OpenSSL Dynamic Engine を移行する ... 873
キーストレージプロバイダー (KSP) の移行 ... 877
JCE プロバイダーを移行する ... 885
クライアント SDK 5 ... 897
最新の SDK の利点 ... 898
サポートされているプラットフォーム ... 899
PKCS #11 ライブラリ ... 902
OpenSSL Dynamic Engine ... 960
OpenSSL プロバイダー ... 968
キーストレージプロバイダー (KSP) ... 976
JCE プロバイダー ... 1007
以前のバージョン ... 1049
クライアント SDK 3 をアップグレードする ... 1050
サポートされているプラットフォーム ... 1059
PKCS #11 ライブラリ ... 1061
OpenSSL Dynamic Engine ... 1104

JCE プロバイダー 1108
KSP および CNG プロバイダー 1139
サードパーティアプリケーションの統合 1152
SSL/TLS のオフロード 1152
仕組み 1153
OpenSSL エンジンを使用した Linux でのオフロード 1155
OpenSSL プロバイダーを使用した Linux でのオフロード 1203
Linux で JSSE を使用したオフロード 1239
Windows でのオフロード 1250
ロードバランサーを追加する (オプション) 1265
Windows Server CA 1272
Windows Server CA を使用するクライアント SDK 5 1272
Windows Server CA を使用するクライアント SDK 3 1277
Oracle Database 暗号化 1283
前提条件の設定 1284
ステップ 2: Oracle データベース設定を更新する 1285
ステップ 3: Oracle TDE マスター暗号化キーの生成 1286
Microsoft SignTool 1288
Microsoft SignTool を使用したクライアント SDK 5 1289
Microsoft SignTool を使用したクライアント SDK 3 1293
Java キーツールとJarsigner 1297
クライアント SDK 5 を使用して Java Keytool および Jarsigner と統合する 1298
クライアント SDK 3 を使用して Java Keytool および Jarsigner と統合する 1312
Microsoft Manifest Generation and Editing Tool 1329
ステップ 1: 前提条件の設定 1329
ステップ 2: 署名用証明書を作成する 1330
ステップ 3: ファイルに署名する 1332
その他のサードパーティベンダーとの統合 1333
モニタリング 1335
クライアント SDK ログ 1335
クライアント SDK 5 ログ記録 1336
クライアント SDK 3 ログ記録 1337
AWS CloudTrail 1339
AWS CloudHSMCloudTrail の情報 1340
AWS CloudHSMログファイルエントリについて 1341
AWS CloudHSMCloudTrail の管理イベント 1342

AWS CloudHSMCloudTrail のサービスイベント 1342
監査ログ 1343
ログ記録の仕組み 1344
ログの表示 1344
ログの解釈 1347
ログリファレンス 1362
CloudWatch メトリクス 1365
パフォーマンス 1367
パフォーマンスデータ 1367
HSM スロットリング 1368
セキュリティ 1369
IAM ポリシーによる API アクセスコントロール 1370
IAM ポリシーを IPv6 にアップグレードする 1370
データ保護 1373
保管中の暗号化 1374
転送中の暗号化 1375
エンドツーエンドの暗号化 1375
クラスターのバックアップ 1376
ID とアクセス管理 1377
IAM ポリシーを使用したアクセス権限の付与 1378
の API アクション AWS CloudHSM 1379
の条件キー AWS CloudHSM 1380
の事前定義された AWS 管理ポリシー AWS CloudHSM 1380
のカスタマー管理ポリシー AWS CloudHSM 1380
サービスリンクロール 1384
コンプライアンス検証 1386
PCI-PIN に関するよくある質問 1388
非推奨 1389
耐障害性 1391
インフラストラクチャセキュリティ 1391
ネットワークの隔離 1391
ユーザーの承認 1392
VPC エンドポイント (AWS PrivateLink) 1392
AWS CloudHSM VPC エンドポイントに関する考慮事項 1392
のインターフェイス VPC エンドポイントの作成 AWS CloudHSM 1392
の VPC エンドポイントポリシーの作成 AWS CloudHSM 1393

更新管理 1394
トラブルシューティング 1395
AWS CloudHSM 既知の問題 1395
すべての HSM インスタンスの既知の問題 1396
hsm1.medium の既知の問題 1403
hsm2m.medium の既知の問題 1403
PKCS#11 ライブラリの既知の問題 1407
JCE SDK の既知の問題 1414
OpenSSL Dynamic Engine SDK の既知の問題 1420
キーストレージプロバイダー (KSP) の既知の問題 1423
OpenSSL プロバイダーの既知の問題 1425
CloudHSM CLI の既知の問題 1426
Amazon Linux 2 を実行する Amazon EC2 インスタンスに関する既知の問題 1427
サードパーティアプリケーションの統合の既知の問題 1428
クラスターの変更に関する既知の問題 1428
hsm2.medium での AWS CloudHSM クライアントバージョン 5.12.0 を使用したオペレーション失敗の既知の問題 1429
クライアント SDK 3 キー同期の失敗 1430
クライアント SDK 3 のパフォーマンス検証 1431
レコメンデーションをテストする 1432
pkpspeed ツールの設定可能なオプション 1433
pkpspeed ツールで実行できるテスト 1433
例 1434
クライアント SDK 5 ユーザーまたはポリシーに整合性のない値が含まれている 1438
一貫性のないユーザー値を解決する 1438
一貫性のない mTLS ポリシー値を解決する 1445
クライアント SDK 5 のユーザーレプリケートの失敗 1446
問題: 選択したユーザーがクラスター全体で同期されない。 1447
問題: 宛先クラスターに、属性が異なるユーザーが存在する 1448
問題: hsm2m.medium から hsm1.medium へのユーザーレプリケートが失敗する 1448
クライアント SDK 5 キーのレプリケートの失敗 1449
問題: 選択したキーがクラスター全体で同期されない。 1449
問題: 同じリファレンスのキーが、宛先クラスターに存在するが、情報または属性が異なる 1451
AWS CloudHSM キーの可用性チェック中にエラーが表示される 1451
JCE によるキーの抽出 1452

GetEncoded、GetPrivateExponent、または getS が null を返します 1452
getEncoded、GetPrivateExponent、または getS は HSM の外部でキーバイトを返します 1453
HSM スロットリング 1453
解決策 1454
接続の消失 1455
CloudWatch に AWS CloudHSM 監査ログがない 1458
非準拠 AES キーラップ 1458
コードが回復不可能なラップされたキーを生成する可能性の判断 1458
コードが回復不可能なラップされたキーを生成する場合に実行が必要なアクション 1460
AWS CloudHSM クラスター作成の失敗の解決 1461
不足しているアクセス権限の追加 1461
サービスにリンクされたロールを手動で作成する 1462
非フェデレーションユーザーの使用 1462
AWS CloudHSM クライアント設定ログの取得 1463
(クライアント SDK 5 対応ツール) 1463
(クライアント SDK 3 対応ツール) 1466
クォータ 1467
ダウンロード 1469
最新のリリース 1469
クライアント SDK 5 リリース: バージョン 5.18.0 1470
以前のリリース 1479
非推奨のリリース 1539
非推奨のクライアント SDK 5 リリース 1539
非推奨のクライアント SDK 3 リリース 1554
サポート終了のリリース 1564
ドキュメント履歴 1565
最新の更新 1565
以前の更新 1573
..... mdlxxv

とは AWS CloudHSM

AWS CloudHSM は、 AWS クラウドの利点とハードウェアセキュリティモジュール (HSMs。ハードウェアセキュリティモジュール (HSM) は、暗号化オペレーションを処理し、暗号化キーの安全なストレージを提供するコンピューティングデバイスです。を使用すると AWS CloudHSM、AWS クラウドにあり、低レイテンシーのアクセスと、HSMs 管理 (バックアップ、プロビジョニング、設定、メンテナンスを含む) を自動化する安全な信頼ルートを持つ高可用性 HSM を完全に制御できます。

AWS CloudHSM は、お客様にさまざまな利点を提供します。

FIPS クラスターと非 FIPS クラスターへのアクセス

AWS CloudHSM は、FIPS と非 FIPS の 2 つのモードでクラスターを提供します。FIPS モードでは、連邦情報処理標準 (FIPS) で検証されたキーとアルゴリズムのみを使用できます。非 FIPS モードは、FIPS の承認に関係なく AWS CloudHSM、 でサポートされているすべてのキーとアルゴリズムを提供します。詳細については、「AWS CloudHSM クラスターモード」を参照してください。

HSM は汎用のシングル テナントであり、FIPS モードのクラスターに対して FIPS 140-2 レベル 3 または FIPS 140-3 レベル 3 のいずれかで検証されています。。

AWS CloudHSM は、アプリケーション用に事前定義されたアルゴリズムとキー長を持つフルマネージド AWS サービスと比較して、より高い柔軟性を提供する汎用 HSMs を使用します。標準に準拠したシングルテナントで、FIPS モードのクラスターに対して FIPS 140-2 レベル 3 または FIPS 140-3 レベル 3 で検証された HSM を提供します。FIPS 140-2 または FIPS 140-3 レベル-3 検証の制限を超えるユースケースを持つお客様のために、 AWS CloudHSM は非 FIPS モードのクラスターも提供します。詳細については「AWS CloudHSM クラスター」を参照してください。

E2E 暗号化は AWS には表示されません。

データプレーンはエンドツーエンド (E2E) で暗号化されており、AWS には表示されないため、自身のユーザー管理 (IAM ロール外) を制御できます。このコントロールのトレードオフは、マネージド型の AWS サービスを使用した場合よりも責任が大きくなることです。

キー、アルゴリズム、アプリケーション開発を完全に制御できます。

AWS CloudHSM では、使用するアルゴリズムとキーを完全に制御できます。暗号化キー (セッションキー、トークンキー、対称キー、非対称キーペアを含む) の生成、保存、インポート、エクスポート、管理、使用ができます。さらに、 AWS CloudHSM SDKs を使用すると、アプリ

ケーション開発、アプリケーション言語、スレッド、アプリケーションが物理的に存在する場所を完全に制御できます。

暗号化ワークロードをクラウドに移行します。

Public Key Cryptography Standards #11 (PKCS #11)、Java Cryptographic Extension (JCE)、Cryptography API: Next Generation (CNG)、または Key Storage Provider (KSP) を使用するパブリックキーインフラストラクチャを移行するお客様は、アプリケーションの変更を少なく AWS CloudHSM して に移行できます。

でできることの詳細については AWS CloudHSM、以下のトピックを参照してください。の使用を開始する準備ができたら AWS CloudHSM、「」を参照してください開始方法。

> **Note**
>
> データの暗号化キーを作成および管理するマネージド型サービスは欲しいが、独自の HSM を運用したくないまたは不要である場合、AWS Key Management Service の使用を検討してください。
> クラウド内の支払い処理アプリケーションの支払い HSM とキーを管理する柔軟性の高いサービスをお探しの場合は、AWS Payment Cryptography の使用を検討してください。

内容

- AWS CloudHSM ユースケース
- の AWS CloudHSM 仕組み
- の料金 AWS CloudHSM

AWS CloudHSM ユースケース

AWS CloudHSM は、さまざまな目標を達成するために使用できます。このトピックのコンテンツでは、 でできることの概要を説明します AWS CloudHSM。

規制の確実な順守

エンタープライズセキュリティ標準に準拠する必要がある企業は、 AWS CloudHSM を使用して、機密性の高いデータを保護するプライベートキーを管理できます。が提供する HSMs AWS CloudHSM は FIPS 140-2 レベル 3 認定を受けており、PCI DSS に準拠しています。さらに、

AWS CloudHSM は PCI PIN に準拠し、PCI-3DS に準拠しています。詳細については、「のコンプライアンス検証 AWS CloudHSM」を参照してください。

データの暗号化と復号

を使用して AWS CloudHSM 、機密性の高いデータ、転送中の暗号化、保管中の暗号化を保護するプライベートキーを管理します。さらに、 は、複数の暗号化 SDKs との標準準拠の統合 AWS CloudHSM を提供します。

文書へのプライベートキーとパブリックキーを使用した署名と検証

暗号化では、プライベートキーを使用して文書に署名すると、受信者はパブリックキーを使用して、他の誰でもないお客様が実際に文書を送信したことを検証できます。を使用して AWS CloudHSM 、この目的のために特別に設計された非対称パブリックキーとプライベートキーのペアを作成します。

HMAC と CMAC を使用したメッセージの認証

暗号化では、Cipher Message Authentication Codes (CMACs) と Hash-based Message Authentication Code (HMAC) を使用して、安全でないネットワークを介して送信されるメッセージを認証し、整合性を確保します。を使用すると AWS CloudHSM、HMACs と CMACs をサポートする対称キーを安全に作成および管理できます。

AWS CloudHSM と の利点を活用する AWS Key Management Service

お客様は AWS CloudHSM と を組み合わせて、キーマテリアルAWS KMSをシングルテナント環境に保存すると同時に、キー管理、スケーリング、クラウド統合の利点を得ることができます AWS KMS。その方法について詳しくは、「AWS Key Management Service 開発者ガイド」の「AWS CloudHSM キーストア」を参照してください。

ウェブサーバーの SSL/TLS 処理のオフロード

インターネット経由でデータを安全に送信するために、ウェブサーバーでは、パブリック/プライベートのキーペアと SSL/TLS パブリックキー証明書を使用して、HTTPS セッションを確立します。このプロセスにはウェブサーバーの多くの計算が含まれますが、その一部を AWS CloudHSM クラスターにオフロードすることで、追加のセキュリティを提供しながら、計算の負

担を軽減できます。で SSL/TLS オフロードを設定する方法については AWS CloudHSM、「」を参照してくださいSSL/TLS のオフロード。

透過的なデータ暗号化 (TDE) の有効化

透過的なデータ暗号化 (TDE) を使用して、データベースファイルを暗号化します。TDE を使用すると、データベースソフトウェアはデータをディスクに保存する前に暗号化します。TDE マスター暗号化キーを AWS CloudHSMの HSM に保存すると、セキュリティを強化するのに役立ちます。で Oracle TDE を設定する方法については AWS CloudHSM、「」を参照してくださいOracle Database 暗号化。

発行認証機関 (CA) のプライベートキーの管理

認証機関 (CA) は、パブリックキーを ID (個人または組織) にバインドするデジタル証明書を発行する信頼されたエンティティです。CA を操作するには、CA によって発行された証明書に署名するプライベートキーを保護して、信頼関係を維持する必要があります。このようなプライベートキーを AWS CloudHSM クラスターに保存し、HSMs を使用して暗号化署名オペレーションを実行できます。

乱数の生成

暗号化キーを作成するための乱数の生成は、オンラインセキュリティの中核です。 は、ユーザーが制御する HSMs で乱数を安全に生成するために使用 AWS CloudHSM でき、ユーザーのみに表示されます。

の AWS CloudHSM 仕組み

このトピックでは、データの安全な暗号化と HSMs での暗号化オペレーションの実行に使用する基本概念とアーキテクチャの概要について説明します。 は、独自の Amazon Virtual Private Cloud (VPC) で AWS CloudHSM 動作します。を使用する前に AWS CloudHSM、まずクラスターを作成し、そのクラスターに HSMs を追加し、ユーザーとキーを作成してから、クライアント SDKs を使用して HSMsをアプリケーションと統合します。これが完了したら、クライアント SDK ログ AWS CloudTrail、監査ログ、Amazon CloudWatch を使用してモニタリング AWS CloudHSMします。

AWS CloudHSMの基本概念と、それらがどのように連携してデータを保護するかについて説明します。

トピック

- AWS CloudHSM クラスター
- のユーザー AWS CloudHSM
- のキー AWS CloudHSM
- SDKs AWS CloudHSM
- AWS CloudHSM クラスターのバックアップ
- でサポートされているリージョン AWS CloudHSM

AWS CloudHSM クラスター

個々の HSMs を同期され、冗長で、可用性の高い方法で連携させることは難しい場合がありますが、クラスターにハードウェアセキュリティモジュール (HSMs) を提供することで AWS CloudHSM 手間がかかります。クラスターは、 が同期 AWS CloudHSM を維持する個々の HSMsのコレクションです。クラスター内にある HSM でタスクまたはオペレーションを行うと、そのクラスター内の他の HSM は、自動的に最新の状態に維持されます。

AWS CloudHSM は、FIPS と非 FIPS の 2 つのモードでクラスターを提供します。FIPS モードでは、連邦情報処理標準 (FIPS) で検証されたキーとアルゴリズムのみを使用できます。非 FIPS モードは、FIPS の承認に関係なく AWS CloudHSM、 でサポートされているすべてのキーとアルゴリズムを提供します。 は、hsm1.medium と hsm2m.medium の 2 種類の HSMs AWS CloudHSM も提供します。各 HSM タイプとクラスターモードの違いの詳細については、「AWS CloudHSM クラスターモード」を参照してください。hsm1.medium HSM タイプはサポートが終了しているため、このタイプでは新しいクラスターを作成できません。詳細については、「非推奨通知」を参照してください。

可用性、耐久性、スケーラビリティの目標を達成するには、複数のアベイラビリティーゾーンにまたがるクラスター内の HSM の数を設定します。1～28 HSMs を持つクラスターを作成できます (デフォルトの制限はAWS 、リージョンごとに AWS アカウントごとに 6 個の HSMsです)。HSMs は、 AWS リージョンの異なるアベイラビリティーゾーンに配置することができます。クラスターに HSM を追加すると、高いパフォーマンスを実現できます。複数のアベイラビリティーゾーンにクラスターを分散すると、冗長性と高可用性を実現します。

クラスターの詳細については、「のクラスター AWS CloudHSM」を参照してください。

クラスターを作成するには、「開始方法」を参照してください。

のユーザー AWS CloudHSM

ほとんどの AWS サービスやリソースとは異なり、 AWS Identity and Access Management (IAM) ユーザーや IAM ポリシーを使用して AWS CloudHSM クラスター内のリソースにアクセスすることはできません。代わりに、 AWS CloudHSM クラスター内の HSM で HSM ユーザーを直接使用します。 HSMs

HSM ユーザーは IAM ユーザーとは異なります。正しい認証情報を持つ IAM ユーザーは、AWS API を介してリソースを操作することで HSM を作成できます。E2E 暗号化は AWS には表示されないため、認証情報は HSM 上で直接行われるため、HSM でのオペレーションを認証するには HSM ユーザー認証情報を使用する必要があります。HSMは、定義および管理する認証情報を使用して、各HSMユーザーを認証します。各 HSM ユーザーには、HSM でユーザーとして実行できるオペレーションを判断する タイプ があります。各HSMは、CloudHSMCLI を使用して定義する認証情報を使用して、各HSMユーザーを認証します。

以前の SDK バージョンシリーズ を使用している場合は、CloudHSM 管理ユーティリティ (CMU) を使用します。

のキー AWS CloudHSM

AWS CloudHSM を使用すると、 AWS CloudHSM クラスター内のシングルテナント HSMs、管理できます。キーは対称でも非対称でもよく、単一セッション用のセッションキー (エフェメラルキー) 、長期使用用のトークンキー (永続キー) でもよく、 AWS CloudHSM キーからエクスポートしてキーにインポートすることもできます。また、一般的な暗号化タスクと関数を完了するためにも使用できます。

- 対称暗号化アルゴリズムと非対称暗号化アルゴリズムの両方を使用して、暗号データ署名と署名検証を行います。
- ハッシュ関数を使用して、メッセージダイジェストと Hash-based Message Authentication Code (HMAC) を計算します。
- 他のキーをラップして保護します。
- 暗号化された安全なランダムデータにアクセスします。

クラスターが持つことができる最大キーは、クラスター内の HSM のタイプによって異なります。例えば、hsm2m.medium には hsm1,medium よりも多くのキーが保存されます。これらの比較については、「AWS CloudHSM クォータ」を参照してください。

さらに、 には、主要な使用と管理に関するいくつかの基本原則 AWS CloudHSM があります。

多くのキータイプとアルゴリズムから選択可能

独自のソリューションをカスタマイズできるように、 はアルゴリズムから選択する多くのキータイプとアルゴリズム AWS CloudHSM を提供し、さまざまなキーサイズをサポートします。詳細については、それぞれ AWS CloudHSM クライアント SDKsオフロード の属性とメカニズムのページを参照してください。

キーの管理方法

AWS CloudHSM キーは SDKsとコマンドラインツールを使用して管理されます。これらのツールを使用してキーを管理する方法については、「のキー AWS CloudHSM」と「のベストプラクティス AWS CloudHSM」を参照してください。

キーの所有者は誰か

では AWS CloudHSM、キーを作成する Crypto User (CU) がキーを所有します。所有者は key share と key unshare コマンドを使用してキーを他の CU と共有または共有解除することができます。詳細については、「CloudHSM CLI を使用してキーを共有または共有解除する」を参照してください。

属性ベースの暗号化によりアクセスと使用を制御可能

AWS CloudHSM では、属性ベースの暗号化を使用できます。これは、キー属性を使用して、ポリシーに基づいてデータを復号できるユーザーを制御できる暗号化の一形式です。

SDKs AWS CloudHSM

を使用する場合は AWS CloudHSM、AWS CloudHSM クライアントソフトウェア開発キット (SDKs) で暗号化オペレーションを実行します。 AWS CloudHSM クライアント SDKsには以下が含まれます。

- 公開鍵暗号規格 #11 (PKCS #11)
- JCE プロバイダー
- OpenSSL Dynamic Engine
- Microsoft Windows のキーストレージプロバイダー (KSP)

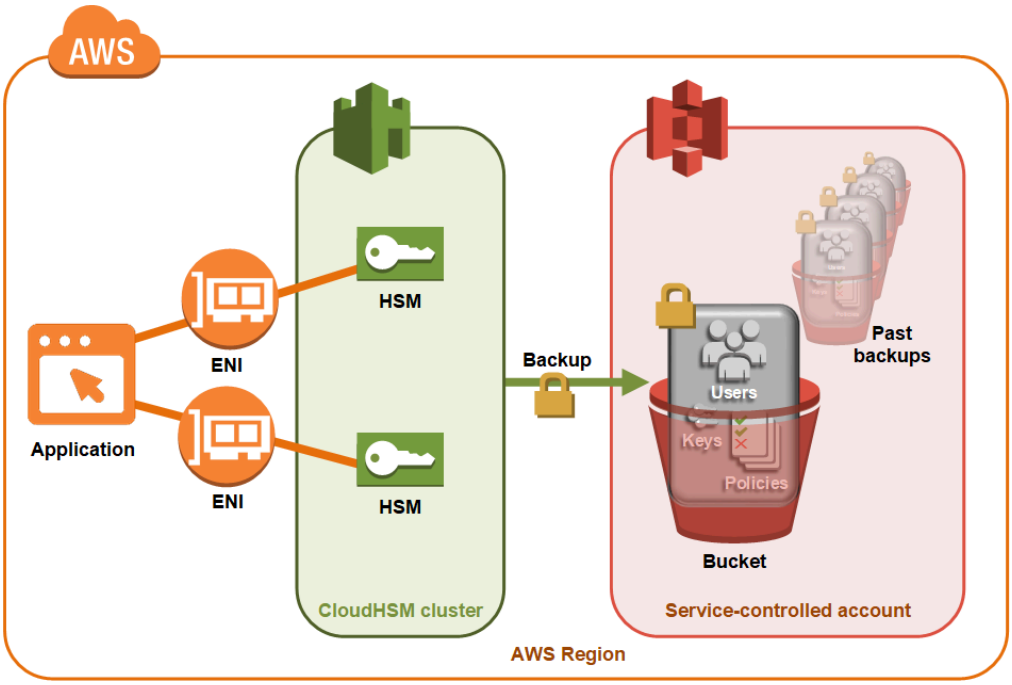
これらの SDK の一部またはすべてを AWS CloudHSM クラスターで使用できます。これらの SDK を使用して HSM で暗号化操作を実行するアプリケーションコードを書き込みます。各 SDK をサポートするプラットフォームと HSM タイプを確認するには、「AWS CloudHSM クライアント SDK 5 がサポートするプラットフォーム」を参照してください。

ユーティリティツールとコマンドラインツールは SDK を使用するだけでなく、アプリケーションの認証情報、ポリシー、設定を構成するためにも必要です。詳細については、AWS CloudHSM コマンドラインツール を参照してください。

クライアント SDK のインストールと使用、あるいはクライアント接続のセキュリティのさらなる詳細については、クライアント SDK と エンドツーエンドの暗号化 を参照してください。

AWS CloudHSM クラスターのバックアップ

AWS CloudHSM は、クラスター内のユーザー、キー、ポリシーを定期的にバックアップします。バックアップは安全で、耐久性が高くて、予測可能なスケジュールで更新されます。下の図は、バックアップとクラスターの関係を示しています。



ユーザーデータのさらなる操作方法の詳細については、クラスターのバックアップ を参照してください。

セキュリティ

が HSM からバックアップ AWS CloudHSM を作成すると、HSM は送信前にすべてのデータを暗号化します AWS CloudHSM。データがプレーンテキスト形式で HSM から外部に出ることはあ

りません。さらに、 はバックアップの復号に使用されるキーにアクセス AWS できない AWS ため、 によるバックアップの復号化はできません。詳細については、クラスターバックアップのセキュリティを参照してください。

耐久性

AWS CloudHSM は、クラスターと同じリージョンのサービス制御の Amazon Simple Storage Service (Amazon S3) バケットにバックアップを保存します。バックアップの耐久レベルは 99.999999999% で、Amazon S3 に保存されているオブジェクトと同じです。

でサポートされているリージョン AWS CloudHSM

でサポートされているリージョンの詳細については AWS CloudHSM、AWS CloudHSM 『』の「リージョンとエンドポイント」、または「リージョンテーブル」を参照してください。 AWS 全般のリファレンス https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

AWS CloudHSM は、特定のリージョンのすべてのアベイラビリティーゾーンで使用できない場合があります。ただし、クラスター内のすべての HSMs 間で AWS CloudHSM 自動的に負荷分散されるため、パフォーマンスには影響しません。

ほとんどの AWS リソースと同様に、クラスターと HSMsはリージョンリソースです。クラスターをリージョン間で再利用したり、拡張したりすることはできません。「の開始方法 AWS CloudHSM」に一覧表示されている必要なステップをすべて実行し、新しいリージョンにクラスターを作成する必要があります。

ディザスタリカバリの目的で、 AWS CloudHSM AWS CloudHSM では、あるリージョンから別のリージョンにクラスターのバックアップをコピーできます。詳細については、「AWS CloudHSM クラスターのバックアップ」を参照してください。

の料金 AWS CloudHSM

では AWS CloudHSM、長期契約や前払いなしで時間単位で支払います。詳細については、 AWS ウェブサイトのAWS CloudHSM 「料金表」を参照してください。

の開始方法 AWS CloudHSM

以下のトピックは、 でクラスターを作成、初期化、アクティブ化するのに役立ちます AWS CloudHSM。これらの手順を完了したら、ユーザーやクラスターを管理できるほか、付属のソフトウェアライブラリを使用して、暗号化オペレーションを実行できるようになります。最適な工クスペリエンスを得るには、一覧表示されている順序でトピックに従ってください。

内容

- の IAM 管理グループを作成する AWS CloudHSM
- の仮想プライベートクラウド (VPC) を作成する AWS CloudHSM
- でクラスターを作成する AWS CloudHSM
- でクラスターのセキュリティグループを確認する AWS CloudHSM
- とやり取りするための Amazon EC2 クライアントインスタンスを起動する AWS CloudHSM
- のクライアント Amazon EC2 インスタンスセキュリティグループを設定する AWS CloudHSM
- で HSM を作成する AWS CloudHSM
- でクラスターの HSM のアイデンティティと信頼性を検証する AWS CloudHSM (オプション)
- でクラスターを初期化する AWS CloudHSM
- CloudHSM CLI をインストールして設定する
- でクラスターをアクティブ化する AWS CloudHSM
- クライアントと の間の相互 TLS を設定する AWS CloudHSM (推奨)
- でキーを作成して使用する AWS CloudHSM

の IAM 管理グループを作成する AWS CloudHSM

の使用を開始する最初のステップ AWS CloudHSM は、IAM アクセス許可を設定することです。

ベストプラクティスとして、 を使用して AWSを AWS アカウントのルートユーザー 操作しないでください AWS CloudHSM。代わりに、 AWS Identity and Access Management (IAM) を使用して IAM ユーザー、IAM ロール、またはフェデレーティッドユーザーを作成します。セクション「IAM ユーザーおよび管理者グループの作成」の手順に従って管理者グループを作成し、[AdministratorAccess] ポリシーをそれにアタッチします。次に新しい管理者ユーザーを作成し、ユーザーをグループに追加します。必要に応じて、追加のユーザーをグループに追加します。追加した各ユーザーは、グループから [AdministratorAccess] ポリシーを継承します。

もう 1 つのベストプラクティスは、実行に必要なアクセス許可のみを持つ AWS CloudHSM 管理者グループを作成することです AWS CloudHSM。必要に応じて個々のユーザーをこのグループに追加します。 AWS へのフルアクセスではなく、グループにアタッチされた制限付きのアクセス許可が各ユーザーに継承されます。以下のカスタマー管理ポリシー AWS CloudHSMセクションには、AWS CloudHSM 管理者グループにアタッチする必要があるポリシーが含まれています。

AWS CloudHSM は、 AWS アカウントのサービスにリンクされたロールを定義します。サービスにリンクされたロールは現在、アカウントが AWS CloudHSM イベントを記録できるようにするアクセス許可を定義します。ロールは、 によって自動的に作成 AWS CloudHSM することも、手動で作成することもできます。ロールを編集することはできませんが、削除することはできます。詳細については、「のサービスにリンクされたロール AWS CloudHSM」を参照してください。

IAM ユーザーおよび管理者グループの作成

にサインアップする AWS アカウント

の使用を開始するには AWS、 が必要です AWS アカウント。の作成の詳細については AWS アカウント、 AWS アカウント管理 リファレンスガイドの「 の開始方法 AWS アカウント」を参照してください。

IAM ユーザーグループにアタッチできる のポリシーの例については、 AWS CloudHSM 「」を参照してくださいの ID とアクセスの管理 AWS CloudHSM。

の仮想プライベートクラウド (VPC) を作成する AWS CloudHSM

のクラスターには Virtual Private Cloud (VPC) が必要です AWS CloudHSM。まだ VPC を作成していない場合は、このトピックのステップに従って、VPC を作成します。

> **Note**
>
> これらの手順に従うと、パブリックサブネットとプライベートサブネットが作成されます。

VPC を作成するには

1. Amazon VPC コンソールの https://console.aws.amazon.com/vpc/ を開いてください。
2. ナビゲーションバーで、リージョンセレクタを使用して、 AWSAWS CloudHSM が現在サポートされているリージョンのいずれかを選択します。

3. VPC を作成 ボタンを選択します。
4. [Resources to create] (作成するリソース) で、[VPC and more] (VPC など) を選択します。
5. [名前タグの自動生成] に、「**CloudHSM**」などの識別可能な名前を入力します。
6. IPv6 CIDR ブロックの場合は、Amazon が提供する IPv6 CIDR ブロックを選択して HSM に IPv6 接続を使用し、クラスターに IPv6 CIDR ブロックを AWS 割り当てます。 HSMs この設定は、デュアルスタックのネットワークタイプをサポートします。IPv6 接続が必要ない場合は、デフォルト設定のままにします。
7. 他のすべてのオプションはデフォルト設定のままにします。
8. VPC を作成 を選択します。
9. VPC が作成されたら、VPC を表示 を選択して、先ほど作成した VPC を表示します。

でクラスターを作成する AWS CloudHSM

クラスターは、個々のハードウェアセキュリティモジュール (HSMs。 AWS CloudHSM は、論理単位として機能するように各クラスターの HSMs を同期します。 は、hsm1.medium と hsm2m.medium の 2 種類の HSMs AWS CloudHSM を提供します。クラスターを作成するときに、2 つのうちのどちらをクラスターに含めるかを選択します。各 HSM タイプとクラスターモードの違いの詳細については、「AWS CloudHSM クラスターモード」を参照してください。

クラスターを作成すると、 はユーザーに代わってクラスターのセキュリティグループ AWS CloudHSM を作成します。このセキュリティグループは、クラスター内の HSM へのネットワークアクセスを制御します。これにより、セキュリティグループの Amazon Elastic Compute Cloud (Amazon EC2) インスタンスからのみインバウンド接続ができます。デフォルトでは、セキュリティグループにインスタンスは一切含まれていません。後で、クライアントインスタンスを起動し、クラスターのセキュリティグループを設定して、HSM との接続および通信を可能にします。

考慮事項

- 以下は、 AWS CloudHSMでクラスターを作成する際の考慮事項です。
  - クラスターを作成すると、 は AWSServiceRoleForCloudHSM という名前のサービスにリンクされたロール AWS CloudHSM を作成します。がロールを作成 AWS CloudHSM できない場合、またはロールがまだ存在しない場合は、クラスターを作成できない可能性があります。詳細については、「AWS CloudHSM クラスター作成の失敗の解決」を参照してください。サービスにリンクされたロールの詳細については、「のサービスにリンクされたロール AWS CloudHSM」を参照してください。

- AWS CloudHSM デュアルスタックエンドポイント (cloudhsmv2.*<region>*.api.aws など) を使用している場合は、IAM ポリシーが更新されて IPv6 が処理されていることを確認します。詳細については、「Control API access with IAM policies」を参照してください。

AWS CloudHSM コンソール、AWS Command Line Interface (AWS CLI)、または AWS CloudHSM API からクラスターを作成できます。

クラスター引数と APIs「」を参照してください。 create-cluster AWS CLI

Console

クラスターを作成するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. ナビゲーションバーで、リージョンセレクタを使用して、AWS CloudHSM が現在サポートされているAWS リージョンのいずれかを選択します。
3. [クラスターを作成] を選択します。
4. [Cluster configuration] セクションで、以下の操作を実行します。

   a. VPC では、の仮想プライベートクラウド (VPC) を作成する AWS CloudHSM で作成した VPC を選択します。

   b. Availability Zone(s) では、各アベイラビリティーゾーンの横にある、作成したプライベートサブネットを選択します。

   > **Note**
   >
   > が特定のアベイラビリティーゾーンでサポート AWS CloudHSM されていない場合でも、クラスター内のすべての HSMs 間で自動的に負荷分散されるため AWS CloudHSM 、パフォーマンスには影響しません。アベイラビリティーゾーンのサポートについてはAWS 全般のリファレンス、AWS CloudHSM 「」の「リージョンとエンドポイント」を参照してください AWS CloudHSM。

   c. H[SM タイプ] では、クラスター内に作成できる HSM タイプとクラスター使用するモードを選択します。各リージョンでサポートされている HSM タイプについては、AWS CloudHSM 料金計算ツール をご覧ください。

> **Important**
>
> クラスターの作成後は、クラスターモードを変更できません。どのタイプとモードがユースケースに適しているかについては、AWS CloudHSM クラスターモード を参照してください。

d. [ネットワークタイプ] では、HSM にアクセスするための IP アドレスプロトコルを選択します。IPv4 を選択すると、アプリケーションと HSM 間の通信は IPv4 のみに制限されます。これはデフォルトのオプションです。デュアルスタックを選択すると、IPv4 と IPv6 の両方の通信が有効になります。デュアルスタックを使用するには、VPC とサブネットの設定に IPv4 と IPv6 の CIDR の両方を追加します。ネットワークタイプは初期設定後に変更するのが難しいです。変更するには、既存クラスターのバックアップを作成し、目的のネットワークタイプを指定した新しいクラスターへ復元します。詳細については、「バックアップからの AWS CloudHSM クラスターの作成」を参照してください。

e. [クラスターモード] では、新しいクラスターを作成するか、既存のバックアップから復元するかを指定します。

- 非 FIPS モードのクラスターのバックアップは、非 FIPS モードのクラスターを復元するためにのみ使用できます。
- FIPS モードのクラスターのバックアップは、FIPS モードのクラスターを復元するためにのみ使用できます。

5. [次へ] を選択します。
6. サービスがバックアップを保持する期間を指定します。

- デフォルトの保存期間である 90 日を受け入れるか、7 ~ 379 日の間に新しい値を入力します。このサービスは、ここで指定した値よりも古いこのクラスター内のバックアップを自動的に削除します。これは後で変更できます。詳細については、「バックアップ保持の設定」を参照してください。

7. [Next] (次へ) を選択します。
8. (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。クラスターに複数のタグを追加するには、タグの追加 を選択します。
9. [Review] (レビュー) を選択します。
10. クラスター設定を確認し、[Create cluster (クラスターの作成)] を選択します。

クラスターの作成が失敗した場合、 AWS CloudHSM サービスにリンクされたロールの問題に関連している可能性があります。障害を解決するためのヘルプについては、「AWS CloudHSM クラスター作成の失敗の解決」を参照してください。

AWS CLI

クラスターを作成するには (AWS CLI)

- コマンドラインプロンプトで、create-cluster コマンドを実行します。HSM インスタンスタイプ、バックアップ保持期間、HSM を作成するサブネットのサブネット ID を指定します。作成したプライベートサブネットのサブネット ID を使用します。サブネットは、アベイラビリティーゾーンごとに 1 つだけ指定できます。

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
                    --backup-retention-policy Type=DAYS,Value=<number of days> \
                    --subnet-ids <subnet ID> \
                    --mode <FIPS> \
                    --network-type <IPV4>

{
    "Cluster": {
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
        "VpcId": "vpc-50ae0636",
        "SubnetMapping": {
            "us-west-2b": "subnet-49a1bc00",
            "us-west-2c": "subnet-6f950334",
            "us-west-2a": "subnet-fd54af9b"
        },
        "SecurityGroup": "sg-6cb2c216",
        "HsmType": "hsm2m.medium",
        "NetworkType": "IPV4",
        "Certificates": {},
        "State": "CREATE_IN_PROGRESS",
        "Hsms": [],
        "ClusterId": "cluster-igklspoyj5v",
        "ClusterMode": "FIPS",
        "CreateTimestamp": 1502423370.069
    }
```

```
}
```

> **Note**
>
> ClusterMode は、hsm1.medium を除くすべての hsm タイプに必要なパラメータです。--mode:
>
> ```
> $ aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
>       --backup-retention-policy Type=DAYS,Value=<number of days> \
>       --subnet-ids <subnet ID> \
>     --mode NON_FIPS
> ```

クラスターの作成が失敗した場合、 AWS CloudHSM サービスにリンクされたロールの問題に関連している可能性があります。障害を解決するためのヘルプについては、「AWS CloudHSM クラスター作成の失敗の解決」を参照してください。

AWS CloudHSM API

クラスターを作成するには (AWS CloudHSM API)

- CreateCluster リクエストを送信します。HSM インスタンスタイプ、バックアップ保持ポリシー、HSM を作成するサブネットのサブネット ID を指定します。作成したプライベートサブネットのサブネット ID を使用します。サブネットは、アベイラビリティーゾーンごとに 1 つだけ指定できます。

クラスターの作成が失敗した場合、 AWS CloudHSM サービスにリンクされたロールの問題に関連している可能性があります。障害を解決するためのヘルプについては、「AWS CloudHSM クラスター作成の失敗の解決」を参照してください。

でクラスターのセキュリティグループを確認する AWS CloudHSM

クラスターを作成するか、HSM をクラスターに追加すると、 は名前を持つセキュリティグループをまだ存在しないcloudhsm-cluster-*<clusterID>*-sg場合は AWS CloudHSM 作成します。このクラスターセキュリティグループには、ポート 2,223～2,225 経由のインバウンド通信とアウトバウンド通信を許可する事前に設定された TCP ルールが含まれます。この SG により、EC2 インスタンスは VPC を使用してクラスター内の HSM と通信できます。

Warning

- 事前設定された TCP ルールを削除または変更しないでください。このルールは、クラスターセキュリティグループに追加されています。このルールによって、接続の問題と HSM への不正アクセスを防ぐことができます。
- クラスターのセキュリティグループに追加することで、HSM への不正アクセスを防ぐことができます。セキュリティグループ内のインスタンスにアクセスできるユーザーはいずれも、HSM にアクセスできます。ほとんどのオペレーションでは、ユーザーは HSM にログインする必要があります。ただし、認証せずに HSM をゼロ化することもできます。ゼロ化すると、キーマテリアル、証明書などのデータは破棄されます。ゼロ化した場合、最後にバックアップしてから作成または変更したデータは失われ、復旧することはできません。不正アクセスを防ぐために、デフォルトのセキュリティグループのインスタンスの変更またはアクセスは、信頼されている管理者のみ行うことができることを確認します。
  - hsm2m.medium クラスターには、権限のないユーザーがクラスターに接続することを制限する mTLS 機能が導入されています。許可されていないユーザーは、ゼロ化を試みる前にクラスターに正常に接続するために有効な mTLS 認証情報が必要です。

次のステップでは、Amazon EC2 インスタンスを起動 し、そのインスタンスに クラスターセキュリティグループを接続 して、HSM に接続します。

とやり取りするための Amazon EC2 クライアントインスタンスを起動する AWS CloudHSM

AWS CloudHSM クラスターと HSM インスタンスを操作して管理するには、HSMs の Elastic Network Interface と通信できる必要があります。これを行う最も簡単な方法として、同じ VPC 内の EC2 インスタンスをクラスターとして使用できます。以下の AWS リソースを使用してクラスターに接続することもできます。

- Amazon VPC ピアリング
- Direct Connect
- VPN 接続

> **Note**
>
> このガイドでは、EC2 インスタンスを AWS CloudHSM クラスターに接続する方法の簡単な例を示します。安全なネットワーク設定に関するベストプラクティスについては、「クラスターへの安全なアクセス」を参照してください。

この AWS CloudHSM ドキュメントでは、通常、クラスターを作成するのと同じ VPC とアベイラビリティーゾーン (AZ) で EC2 インスタンスを使用していることを前提としています。

EC2 インスタンスを作成するには

1. https://console.aws.amazon.com/ec2/ で EC2 ダッシュボード を開きます。
2. [Launch instance] (インスタンスを起動) を選択します。ドロップダウンメニューから、[インスタンスの起動] を選択します。
3. [名前] フィールドに、EC2 インスタンスの名前を入力します。
4. [アプリケーションおよび OS イメージ (Amazon マシンイメージ)] セクションで、CloudHSM がサポートするプラットフォームに対応する Amazon マシンイメージ (AMI) を選択します。詳細については、「AWS CloudHSM クライアント SDK 5 がサポートするプラットフォーム」を参照してください。
5. インスタンスタイプ] セクションで、インスタンスタイプを選択します。
6. [キーペア] セクションで、既存のキーペアを使用するか、[新しいキーペアの作成] を選択して次の手順を実行します。
   a. [キーペア名] に、新しいキーペアの名前を入力します。
   b. [キーペアタイプ] で、キーペアタイプを選択します。
   c. [プライベートキーファイル形式] で、プライベートキーファイルの形式を選択します。
   d. [Create key pair] (キーペアを作成) を選択します。
   e. プライベートキーファイルをダウンロードして保存します。

> **Important**
>
> このタイミングでのみ、プライベートキーファイルを保存できます。ダウンロードしたファイルを安全な場所に保存します。インスタンスの起動時に、キーペアの名前を指

定する必要があります。さらに、インスタンスに接続するたびに対応するプライベートキーを提供し、セットアップ時に作成したキーペアを選択する必要があります。

7. [ネットワーク設定] で [編集] を選択します。
8. VPC には、クラスター用に以前に作成した VPC を選択します。
9. [サブネット] で、VPC で以前に作成したパブリックサブネットを選択します。
10. [自動割り当てパブリック IP] で、[有効] を選択します。
11. [自動割り当て IPv6 IP] では、[有効] を選択すると、クラスターで IPv6 接続を使用できるようになり、デュアルスタックのネットワークタイプを利用できます。このオプションを有効にする場合は、Amazon EC2 インスタンスから HSM への IPv6 送信トラフィックを許可するため、このインスタンスのセキュリティグループのルール、VPC およびサブネットのルートテーブル、ネットワーク ACL を更新してください。
12. [Select an existing security group (既存のセキュリティグループの選択)] を選択します。
13. [共通セキュリティグループ] では、ドロップダウンメニューからデフォルトのセキュリティグループを選択します。
14. ストレージの設定 では、ドロップダウンメニューを使用してストレージ設定を選択します。
15. 「概要」ウィンドウで「インスタンスを起動」を選択します。

> **Note**
>
> このステップを完了すると、EC2 インスタンスの作成プロセスが開始されます。

Linux Amazon EC2 クライアントを作成する方法の詳細については、Amazon EC2 Linux インスタンスの開始方法 を参照してください。実行中のクライアントへの接続については、次のトピックを参照してください。

- SSH を使用した Linux インスタンスへの接続
- PuTTY を使用した Windows から Linux インスタンスへの接続

Amazon EC2 ユーザーガイドは、Amazon EC2 インスタンスを設定および使用するための詳細な手順が含まれます。次のリストは、Linux と Windows Amazon EC2 クライアントで使用できるドキュメントの概要を示します。

- Linux Amazon EC2 クライアントを作成するには、Amazon EC2 Linux インスタンスの開始方法 を参照してください。

  実行中のクライアントへの接続については、次のトピックを参照してください。

  - SSH を使用した Linux インスタンスへの接続
  - PuTTY を使用した Windows から Linux インスタンスへの接続
- Windows Amazon EC2 クライアントを作成するには、 Amazon EC2 Windows インスタンスの開始方法 を参照してください。Windows クライアントに接続する方法については、「Windows インスタンスへの接続」を参照してください。

> **Note**
>
> EC2 インスタンスは、このガイドに含まれるすべての AWS CLI コマンドを実行できます。AWS CLI がインストールされていない場合は、「AWS Command Line Interface」からダウンロードすることができます。Windows を使用している場合は、64 ビットまたは 32 ビットの Windows インストーラをダウンロードして実行できます。Linux または macOS を使用している場合は、pip を使用して CLI をインストールできます。

のクライアント Amazon EC2 インスタンスセキュリティグループを設定する AWS CloudHSM

でクラスターの Amazon EC2 インスタンスを起動すると AWS CloudHSM、デフォルトの Amazon VPC セキュリティグループに関連付けられます。このトピックでは、クラスターセキュリティグループを EC2 インスタンスに関連付ける方法について説明します。この関連付けにより、EC2 インスタンスで実行されている AWS CloudHSM クライアントが HSMsと通信できるようになります。EC2 インスタンスを AWS CloudHSM クラスターに接続するには、VPC のデフォルトのセキュリティグループを適切に設定し、クラスターのセキュリティグループをインスタンスに関連付ける必要があります。

設定の変更を完了するには、次の手順を実行します。

トピック

- ステップ 1. デフォルトのセキュリティグループの変更
- ステップ 2. Amazon EC2 インスタンスを AWS CloudHSM クラスターに接続する

ステップ 1. デフォルトのセキュリティグループの変更

クライアントソフトウェアをダウンロードしてインストールし、HSM と通信できるように、SSH 接続または RDP 接続が許可されるようにデフォルトのセキュリティグループを変更する必要があります。

デフォルトのセキュリティグループを変更するには

1. https://console.aws.amazon.com/ec2/ で EC2 ダッシュボード を開きます。
2. インスタンス (実行中) を選択し、 AWS CloudHSM クライアントをインストールする EC2 インスタンスの横にあるチェックボックスをオンにします。
3. [セキュリティ] タブで、[デフォルト] という名前のセキュリティグループを選択します。
4. ページの一番上で、[アクション]、[インバウンドのルールの編集] の順に選択します。
5. [Add rule (ルールの追加)] を選択します。
6. [タイプ] で、以下のいずれかを実行します。
   - Windows Server の Amazon EC2 インスタンスで、RDP を選択します。ポート 3389 は自動的に追加されています。
   - Linux Amazon EC2 インスタンスの場合は、SSH を選択します。ポート範囲 22 は自動的に追加されています。
7. いずれのオプションでも、[ソース] を [My IP] に設定すると、Amazon EC2 インスタンスと通信できるようになります。

   > **Important**
   >
   > 誰もがインスタンスにアクセスできないようにするには、CIDR 範囲として 0.0.0.0/0 を指定しないでください。

8. [保存] を選択します。

ステップ 2. Amazon EC2 インスタンスを AWS CloudHSM クラスターに接続する

EC2 インスタンスがクラスター内の HSM と通信できるように、クラスターのセキュリティグループを EC2 インスタンスに接続する必要があります。クラスターのセキュリティグループには、ポート 2223〜2225 経由のインバウンド通信を許可する事前に設定されたルールが含まれます。

EC2 インスタンスを AWS CloudHSM クラスターに接続するには

1. https://console.aws.amazon.com/ec2/ で EC2 ダッシュボード を開きます。
2. インスタンス (実行中) を選択し、 AWS CloudHSM クライアントをインストールする EC2 インスタンスのチェックボックスをオンにします。
3. ページの一番上で、アクション、セキュリティ、セキュリティグループの変更 を選択します。
4. クラスターの ID と一致するグループ名のセキュリティグループ (例: `cloudhsm-cluster-<clusterID>-sg`) を選択します。
5. [セキュリティグループを追加] を選択します。
6. [保存] を選択します。

> **Note**
>
> 最大 5 つのセキュリティグループを 1 つの Amazon EC2 インスタンスに割り当てることができます。上限に達した場合は、Amazon EC2 のインスタンスのデフォルトのセキュリティグループとクラスターのセキュリティグループを変更する必要があります。
>
> デフォルトのセキュリティグループで、以下の操作を行います。
>
> - クラスターセキュリティグループからポート 2223-2225 経由で TCP プロトコルを使用したトラフィックを許可するアウトバウンドルールを追加します。
>
> クラスターのセキュリティグループで、以下の操作を行います。
>
> - デフォルトのセキュリティグループからポート 2223-2225 経由で TCP プロトコルを使用したトラフィックを許可するインバウンドルールを追加します。

で HSM を作成する AWS CloudHSM

でクラスターを作成したら AWS CloudHSM、ハードウェアセキュリティモジュール (HSM) を作成できます。ただし、クラスター内に HSM を作成する前に、クラスターを初期化されていない状態にする必要があります。クラスターの状態を確認するには、 AWS CloudHSM コンソールでクラスターページを表示するか、 AWS CLI を使用して describe-clusters コマンドを実行するか、 AWS CloudHSM API で DescribeClusters リクエストを送信します。HSM は、AWS CloudHSM コンソール、AWS CLI、あるいは AWS CloudHSM API で作成できます。

> **Important**
>
> クラスターが初期化されていない状態にある間は、HSM を 1 つだけ作成します。

Console

HSM を作成するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. HSM を作成するクラスターの ID の横のラジオボタンを選択します。
3. アクション を選択します。ドロップダウンメニューから 初期化 を選択します。
4. 作成中の HSM のアベイラビリティーゾーン (AZ) を選択します。
5. [作成] を選択します。

クラスターと HSM を作成した後、オプションで HSM のアイデンティティを確認するか、直接「クラスターの初期化」に進むことができます。

AWS CLI

HSM を作成するには (AWS CLI)

- コマンドラインプロンプトで、create-hsm コマンドを実行します。以前に作成したクラスターのクラスター ID と、HSM のアベイラビリティーゾーンを指定します。アベイラビリティーゾーンは、us-west-2a、us-west-2b などの形式で指定します。

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-
zone <Availability Zone>

{
    "Hsm": {
        "HsmId": "hsm-ted36yp5b2x",
        "EniIp": "10.0.1.12",
        "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733",
        "AvailabilityZone": "us-west-2a",
        "ClusterId": "cluster-igklspoyj5v",
        "EniId": "eni-5d7ade72",
        "SubnetId": "subnet-fd54af9b",
        "State": "CREATE_IN_PROGRESS"
```

```
    }
}
```

クラスターと HSM を作成した後、オプションで HSM のアイデンティティを確認するか、直接「クラスターの初期化」に進むことができます。

AWS CloudHSM API

HSM (AWS CloudHSM API) を作成するには

- CreateHsm リクエストを送信します。以前に作成したクラスターのクラスター ID と、HSM のアベイラビリティーゾーンを指定します。

クラスターと HSM を作成した後、オプションで HSM のアイデンティティを確認するか、直接「クラスターの初期化」に進むことができます。

でクラスターの HSM のアイデンティティと信頼性を検証する AWS CloudHSM (オプション)

クラスターを初期化するには AWS CloudHSM、クラスターの最初のハードウェアセキュリティモジュール (HSM) によって生成された証明書署名リクエスト (CSR) に署名します。その前に、HSM のアイデンティティと正当性を確認することをお勧めします。

> **Note**
>
> このプロセスはオプションです。ただし、使用できるのはクラスターが初期化される前に限られます。クラスターの初期化後は、このプロセスを使用して証明書の取得や HSM の検証を行うことはできません。

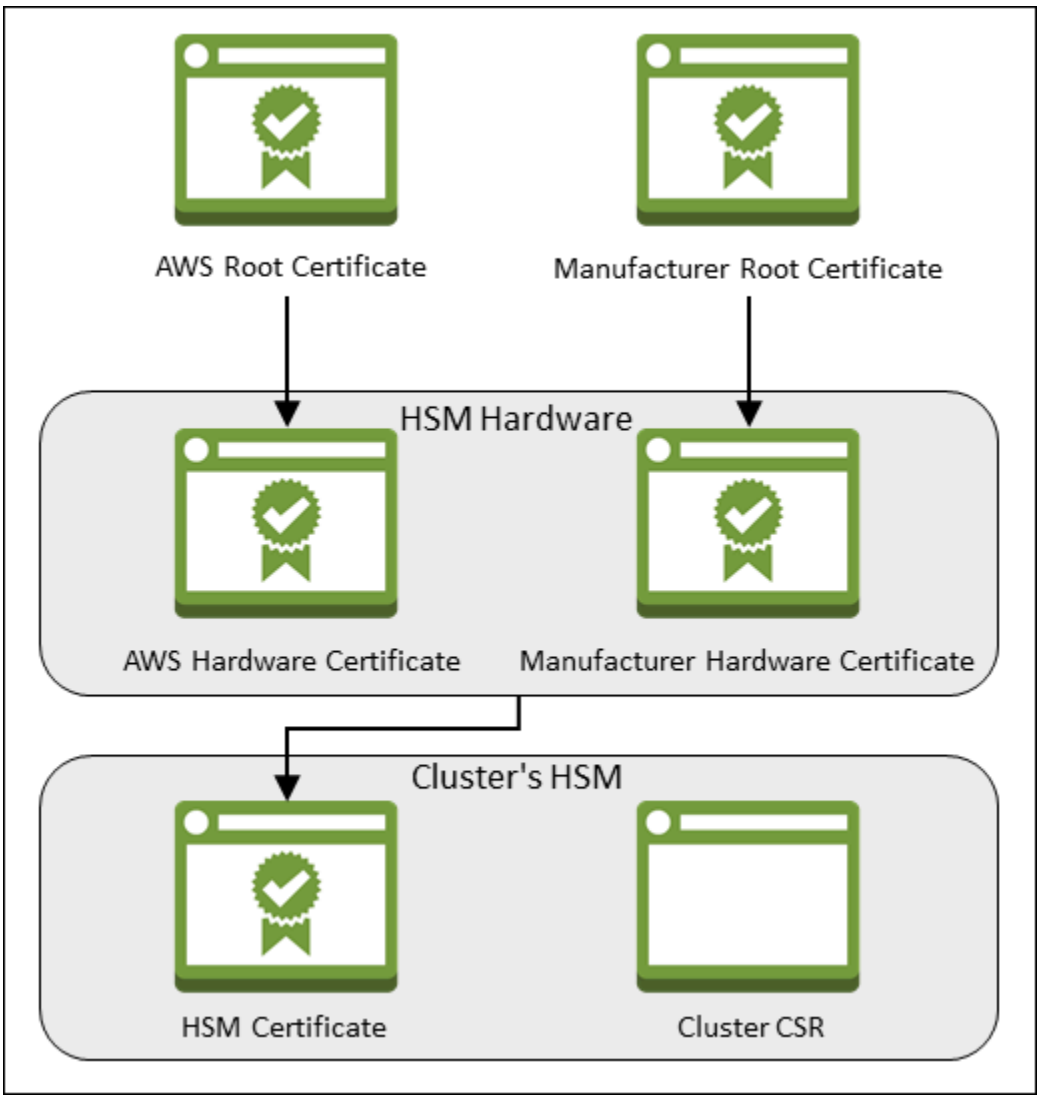
クラスターの最初の HSM のアイデンティティを確認するには、次のステップを実行します。

1. 証明書と CSR の取得 - このステップでは、3 つの証明書と CSR を HSM から取得します。また、2 つのルート証明書も取得します。1 つは HSM ハードウェアメーカーから AWS CloudHSM 、もう 1 つは HSM ハードウェアメーカーから取得します。
2. 証明書チェーンの検証 – このステップでは、2 つの証明書チェーンを作成します。1 つは AWS CloudHSM ルート証明書、もう 1 つは製造元ルート証明書です。次に、これらの証明書チェーン

を使用して HSM 証明書を検証し、 AWS CloudHSM とハードウェア製造元の両方が HSM のアイデンティティと信頼性を証明しているかどうかを確認します。

3. パブリックキーの比較 - このステップでは、HSM 証明書とクラスター CSR のパブリックキーを抽出および比較し、それらが同じであることを確認します。これにより、CSR は認証され、信頼された HSM によって生成されていることを確信できます。

次の図は、CSR、証明書、およびその相互関係を示しています。以下のリストでは、各証明書を定義します。



AWS ルート証明書

これはルート証明書 AWS CloudHSMです。

製造元のルート証明書

これは、ハードウェア製造元のルート証明書です。

AWS ハードウェア証明書

AWS CloudHSM は、HSM ハードウェアがフリートに追加されると、この証明書を作成しました。この証明書は、ハードウェア AWS CloudHSM を所有する をアサートします。

製造元のハードウェア証明書

この証明書は、HSM ハードウェア製造元が HSM ハードウェアを製造したときに作成したものです。この証明書は、製造元がハードウェアを作成したことを主張しています。

HSM 証明書

クラスターの最初の HSM を作成すると、FIPS 検証済みハードウェアが HSM 証明書を生成します。この証明書は、HSM ハードウェアが HSM の作成元であることを証明します。

クラスター CSR

最初の HSM によってクラスター CSR が作成されます。クラスター CSR に署名する と、クラスターがクレームされます。次に、署名した CSR を使用して クラスターを初期化 できます。

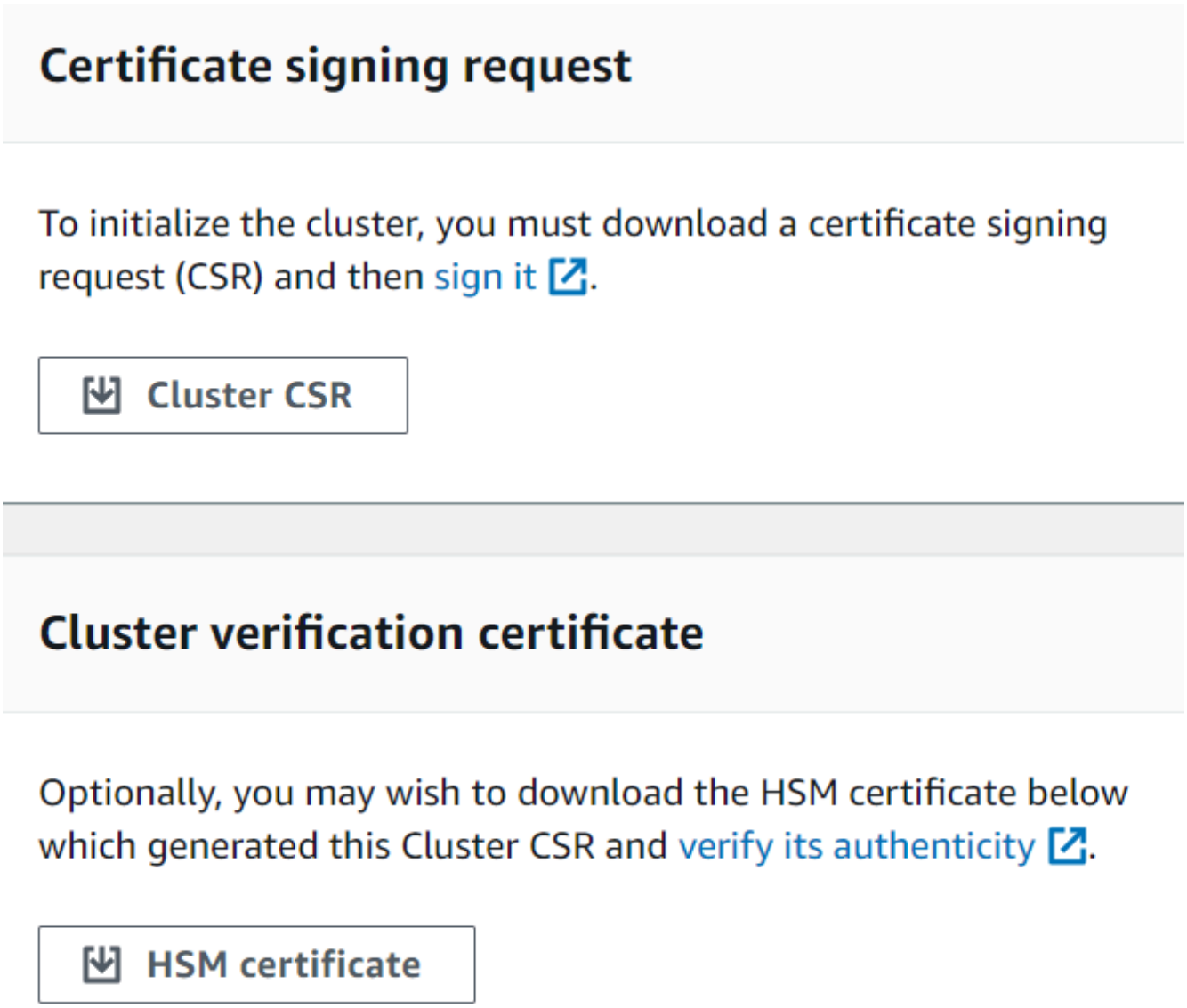
ステップ 1. HSM からの証明書の取得

HSM のアイデンティティと正当性を確認するには、最初に CSR と 5 つの証明書を取得します。HSM から 3 つの証明書を取得します。これは、AWS CloudHSM コンソール、 AWS Command Line Interface (AWS CLI)、または AWS CloudHSM API で実行できます。

Console

CSR および HSM 証明書を取得するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. 検証する HSM のクラスター ID の横にあるラジオボタンをオンにします。
3. アクション を選択します。ドロップダウンメニューから 初期化 を選択します。
4. HSM を作成する 前のステップ を完了していない場合は、作成する HSM のアベイラビリティーゾーン (AZ) を選択します。次に、作成 を選択します。
5. 証明書と CSR の準備ができると、それらをダウンロードするためのリンクが表示されます。



6. 各リンクを選択し、CSR と証明書をダウンロードして保存します。以降のステップを簡素化するために、すべてのファイルを同じディレクトリに保存し、デフォルトのファイル名を使用します。

AWS CLI

CSR および HSM 証明書を取得するには (AWS CLI)

- コマンドプロンプトで、describe-clusters コマンドを 4 回実行し、毎回 CSR と異なる証明書を抽出してファイルに保存します。

a. 次のコマンドを発行してクラスター CSR を抽出します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query
 'Clusters[].Certificates.ClusterCsr' \
                                   > <cluster ID>_ClusterCsr.csr
```

b. 次のコマンドを発行して HSM 証明書を抽出します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query
 'Clusters[].Certificates.HsmCertificate' \
                                   > <cluster ID>_HsmCertificate.crt
```

c. 次のコマンドを発行して、 AWS ハードウェア証明書を抽出します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query
 'Clusters[].Certificates.AwsHardwareCertificate' \
                                   > <cluster ID>_AwsHardwareCertificate.crt
```

d. 次のコマンドを発行して製造元のハードウェア証明書を抽出します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query
 'Clusters[].Certificates.ManufacturerHardwareCertificate' \
                                   > <cluster
 ID>_ManufacturerHardwareCertificate.crt
```

AWS CloudHSM API

CSR 証明書と HSM 証明書を取得するには (AWS CloudHSM API)

- DescribeClusters リクエストを送信し、レスポンスから CSR と証明書を抽出して保存します。

ステップ 2. ルート証明書の取得

AWS CloudHSM および製造元のルート証明書を取得するには、次の手順に従います。ルート証明書ファイルを、CSR と HSM 証明書ファイルが含まれているディレクトリに保存します。

AWS CloudHSM および製造元のルート証明書を取得するには

1. AWS CloudHSM ルート証明書のダウンロード: AWS_CloudHSM_Root-G1.zip
2. HSM タイプに適した製造元のルート証明書をダウンロードします。
   - hsm1.medium 製造元ルート証明書: liquid_security_certificate.zip
   - hsm2m.medium 製造元ルート証明書: liquid_security_certificate.zip

   > **Note**
   >
   > ランディングページから各証明書をダウンロードするには、次のリンクにアクセスします。
   >
   > - hsm1.medium の製造元ルート証明書のランディングページ
   > - hsm2m.medium の製造元ルート証明書のランディングページ
   >
   > [Download Certificate] リンクを右クリックしてから、[Save Link As...] を選択して証明書ファイルを保存することが必要になる場合があります。
3. ファイルをダウンロードした後、内容を抽出 (解凍) します。

ステップ 3. 証明書チェーンの確認

このステップでは、2 つの証明書チェーンを作成します。1 つは AWS CloudHSM ルート証明書、もう 1 つは製造元ルート証明書です。次に、OpenSSL を使用して各証明書チェーンの HSM 証明書を検証します。

証明書チェーンを作成するには、Linux シェルを開きます。OpenSSL が必要です (ほとんどの Linux シェルにあります)。さらに、ダウンロードしたルート証明書と HSM 証明書ファイルが必要です。ただし、このステップ AWS CLI では は必要ありません。シェルを AWS アカウントに関連付ける必要はありません。

AWS CloudHSM ルート証明書を使用して HSM 証明書を検証するには

1. ダウンロードしたルート証明書と HSM 証明書ファイルの保存先のディレクトリに移動します。以下のコマンドでは、すべての証明書が現在のディレクトリにあり、デフォルトのファイル名を使用しているものとします。

   次のコマンドを使用して、ハードウェア証明書と AWS CloudHSM ルート証明書を含む AWS 証明書チェーンをその順序で作成します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

   ```
   $ cat <cluster ID>_AwsHardwareCertificate.crt \
         AWS_CloudHSM_Root-G1.crt \
         > <cluster ID>_AWS_chain.crt
   ```

2. AWS 証明書チェーンで HSM 証明書を検証するには、次の OpenSSL コマンドを使用します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

   ```
   $ openssl verify -CAfile <cluster ID>_AWS_chain.crt <cluster ID>_HsmCertificate.crt
   <cluster ID>_HsmCertificate.crt: OK
   ```

製造元のルート証明書で HSM 証明書を検証するには

1. 次のコマンドを使用して、製造元のハードウェア証明書と、製造元のルート証明書が含まれている証明書チェーンを、その順番で作成します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

   ```
   $ cat <cluster ID>_ManufacturerHardwareCertificate.crt \
         liquid_security_certificate.crt \
   ```

```
    > <cluster ID>_manufacturer_chain.crt
```

2. 製造元の証明書チェーンで HSM 証明書を検証するには、次の OpenSSL コマンドを使用します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

```
$ openssl verify -CAfile <cluster ID>_manufacturer_chain.crt <cluster
 ID>_HsmCertificate.crt
<cluster ID>_HsmCertificate.crt: OK
```

ステップ 4. パブリックキーの抽出と比較

OpenSSL を使用して HSM 証明書とクラスター CSR のパブリックキーを抽出および比較して、それらが同じであることを確認します。

パブリックキーを比較するには、Linux シェルを使用します。ほとんどの Linux シェルで使用できる OpenSSL が必要ですが、このステップ AWS CLI では は必要ありません。シェルを AWS アカウントに関連付ける必要はありません。

パブリックキーを抽出して比較するには

1. 次のコマンドを使用して、HSM 証明書からパブリックキーを抽出します。

```
$ openssl x509 -in <cluster ID>_HsmCertificate.crt -pubkey -noout > <cluster
 ID>_HsmCertificate.pub
```

2. 次のコマンドを使用して、クラスター CSR からパブリックキーを抽出します。

```
$ openssl req -in <cluster ID>_ClusterCsr.csr -pubkey -noout > <cluster
 ID>_ClusterCsr.pub
```

3. 次のコマンドを使用してパブリックキーを比較します。パブリックキーが同じである場合、次のコマンドによる出力はありません。

```
$ diff <cluster ID>_HsmCertificate.pub <cluster ID>_ClusterCsr.pub
```

HSM のアイデンティティと正当性を確認したら、「クラスターの初期化」に進みます。

でクラスターを初期化する AWS CloudHSM

クラスターを作成して にハードウェアセキュリティモジュール (HSM) を追加したら AWS CloudHSM、クラスターを初期化できます。 クラスターを初期化するには、以下のトピックの手順を実行します。

> **Note**
>
> クラスターを初期化する前に、HSM のアイデンティティと正当性を検証するプロセスを確認します。このプロセスはオプションですが、使用期間はクラスターが初期化されるまでの間に限ります。クラスターの初期化後は、このプロセスを使用して証明書の取得や HSM の検証を行うことはできません。

トピック

- 概要
- ステップ 1. クラスター CSR の取得
- ステップ 2. ルート CA のプライベートキーを作成する
- ステップ 3. CSR の署名
- ステップ 4. クラスターの初期化

概要

クラスターの初期化プロセスでは、証明書ベースの認証システムを通じて、クラスターと HSMsの所有権と制御を確立します。このプロセスは、お客様がクラスター内の HSMs の唯一の所有者であることを証明し、HSM への今後のすべての接続に必要な信頼の基盤を作成します HSMs 。

このページでは、以下の方法を示します。

- クラスターの証明書署名リクエスト (CSR) を取得します。
- プライベートキー (複数可) を生成して使用して、自己署名ルート証明書または証明書チェーンを作成します。
- クラスターの CSR に署名して、署名付き HSM 証明書を作成します。
- 署名付き HSM 証明書と自己署名証明書または証明書チェーンを使用してクラスターを初期化します。

始める準備ができたら、「ステップ 1. クラスター CSR の取得」を参照してください。

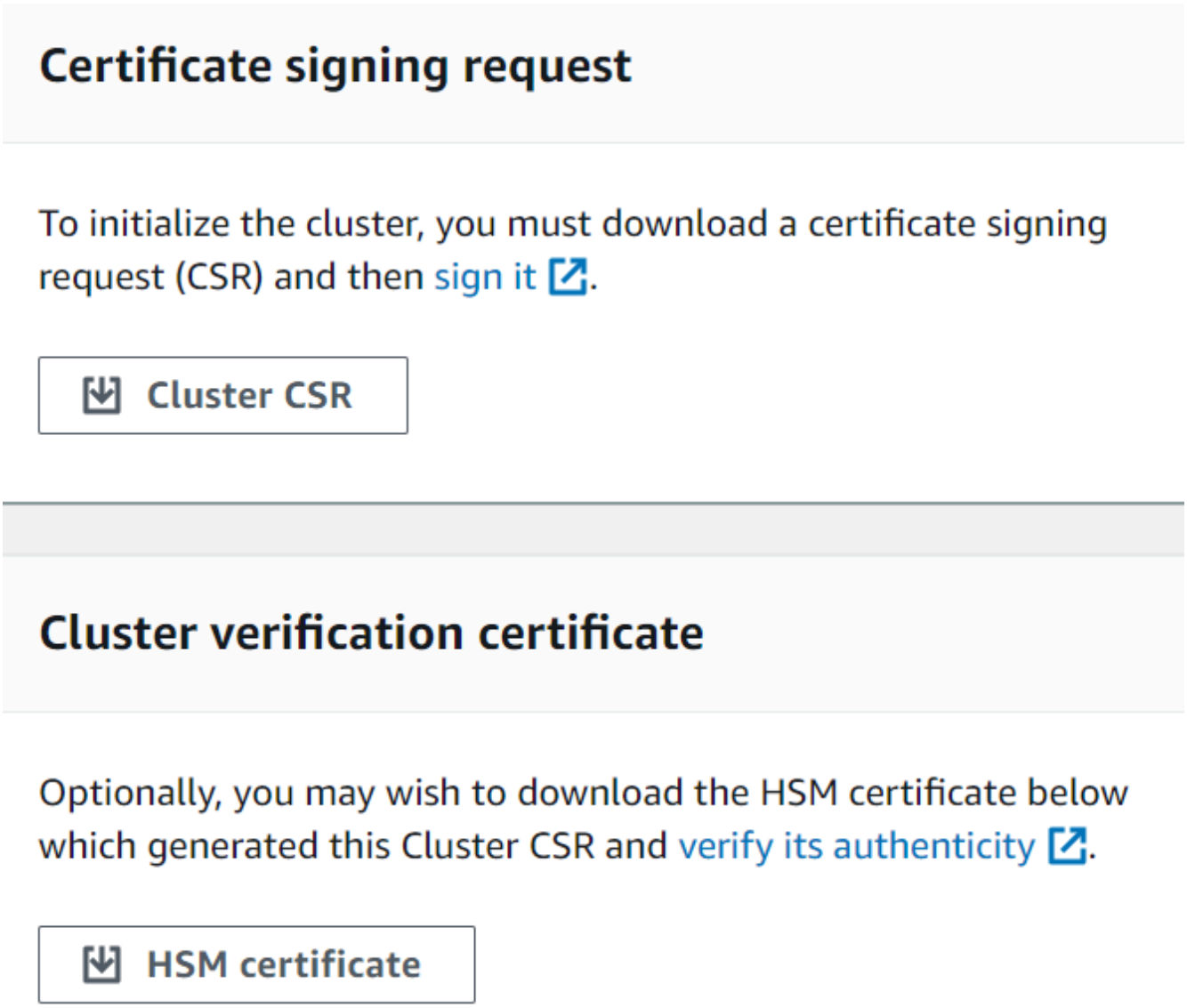
ステップ 1. クラスター CSR の取得

クラスターを初期化する前に、クラスターの最初の HSM によって生成された証明書署名リクエスト (CSR) をダウンロードして署名する必要があります。「クラスターの HSM のアイデンティティの確認」のステップに従っていれば、既に CSR があり、「CSR の署名」に進むことができます。それ以外の場合は、AWS CloudHSM コンソール、 AWS Command Line Interface (AWS CLI)、または AWS CloudHSM API を使用して CSR を取得します。

Console

CSR を取得するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. 検証する HSM のクラスター ID の横にあるラジオボタンをオンにします。
3. アクション を選択します。ドロップダウンメニューから 初期化 を選択します。
4. HSM を作成する 前のステップ を完了していない場合は、作成する HSM のアベイラビリティーゾーン (AZ) を選択します。次に、作成 を選択します。
5. CSR の準備ができると、CSR をダウンロードするためのリンクが表示されます。



6. [Cluster CSR] を選択して CSR をダウンロードし、保存します。

AWS CLI

CSR を取得するには (AWS CLI)

- コマンドプロンプトで、次の describe-clusters コマンドを実行し、CSR を抽出してファイルに保存します。*<cluster ID>* を、前に作成したクラスターの ID に置き換えます。

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query 'Clusters[].Certificates.ClusterCsr'
 \
```

```
                                        > <cluster ID>_ClusterCsr.csr
```

AWS CloudHSM API

CSR (AWS CloudHSM API) を取得するには

1. DescribeClusters リクエストを送信します。
2. レスポンスから CSR を抽出して保存します。

ステップ 2. ルート CA のプライベートキーを作成する

> **Note**
>
> 実稼働用クラスターでは、作成しようとしているキーはランダム性の信頼できるソースを使用して安全な方法で作成されている必要があります。安全なオフサイトあるいはオフライン HSM またはその同等を使用することが推奨されます。キーを安全に保存します。このキーによってクラスターの ID が確立され、クラスターに含まれる HSM をユーザーが単独で制御できるようになります。
> 開発とテストには、クラスター証明書の作成と署名に任意のツール (OpenSSL) などを使用できます。以下の例では、 キーを作成する方法を示します。キーを使用して自己署名証明書 (以下を参照) を作成したら、安全な方法でキーを保存する必要があります。 AWS CloudHSM インスタンスにサインインするには、証明書が存在する必要がありますが、プライベートキーは存在しません。

次の表は、証明書生成でサポートされているアルゴリズム、キーサイズ、曲線の概要を示しています。

アルゴリズム	サイズ/カーブ
RSA PKCSv1.5	2048、3072、4096
RSA-PSS	2048、3072、4096
ECDSA	prime256v1、secp384r1、secp521r1
ダイジェスト	SHA-224、SHA-256、SHA-384、SHA-512

次のコマンド例を使用して、自己署名ルート CA のプライベートキーを作成します。

```
$ openssl genrsa -aes256 -out customerRootCA.key 2048
Generating RSA private key, 2048 bit long modulus
........+++
............+++
e is 65537 (0x10001)
Enter pass phrase for customerRootCA.key:
Verifying - Enter pass phrase for customerRootCA.key:
```

ステップ 3. CSR の署名

前のステップでは、クラスターの CSR を取得し、ルート CA のプライベートキーを作成しました。このステップでは、プライベートキーを使用して署名証明書を生成し、クラスターの CSR に署名します。以下のトピックでは、OpenSSL を使用して単一の自己署名証明書または証明書チェーンを作成するプロセスについて説明します。このステップ AWS CLI では は必要なく、シェルを AWS アカウントに関連付ける必要はありません。

> **Important**
>
> クラスターを初期化するには、トラストアンカーが RFC 5280 に準拠し、次の要件を満たしている必要があります。
>
> - X509v3 エクステンションを使用する場合は、X509v3 基本制約エクステンションが必要です。
> - トラストアンカーは自己署名証明書でなければなりません。
> - エクステンションの値は互いに矛盾してはいけません。

クラスターの CSR に署名するには、次のいずれかの方法を選択します。

証明書のアプローチを選択する

次の 2 つのアプローチのいずれかを選択する必要があります。両方のアプローチを完了しないでください。

オプション A: 単一の自己署名証明書

クラスターの CSR に署名する自己署名ルート証明書を 1 つ作成します。これは、信頼を確立する最もシンプルで直接的な方法です。

推奨対象:

- 外部 PKI が不要な環境
- シンプルさが優先されるテスト環境と開発環境

次に移動します。 自己署名証明書を 1 つ作成する

オプション B: 中間 CA を持つ証明書チェーン

中間認証機関を使用して証明書チェーンを作成します。中間証明書チェーンは、証明書の発行を中間 CAs に委任しながらルート認証機関 (CAs) をオフラインのままにすることで、セキュリティ、スケーラビリティ、柔軟性を強化し、ルート CA が侵害されるリスクを軽減します。

推奨対象:

- 外部 PKI が必要な環境
- AWS Private Certificate Authority (PCA) との統合

AWS PCA 統合の例: AWS プライベート認証機関を使用して中間 CA 証明書を作成および管理できます。これにより、ルート CA をオフラインにしておくというセキュリティ上の利点を維持しながら、更新や取り消しを含む証明書のライフサイクル管理を自動化できます。 AWS PCA の詳細については、AWS Private Certificate Authority ユーザーガイドを参照してください。

次に移動します。 中間認証機関 (ICA) チェーンを作成する

自己署名証明書を 1 つ作成する

本稼働のクラスター用のプライベートキーの作成に使用する信頼できるハードウェアも、このキーを使用した自己署名証明書を生成するソフトウェアツールを提供していることが必要です。次の例では、OpenSSL と前のステップで作成したプライベートキーを使用して、自己署名ルート CA 署名証明書を作成します。証明書の有効期間は 10 年 (3652 日) です。画面の指示を読み、プロンプトに従います。

```
$ openssl req -new -x509 -days 3652 -key customerRootCA.key -out customerRootCA.crt
Enter pass phrase for customerRootCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

このコマンドは customerRootCA.crt という名前の証明書を作成します。この証明書は、 AWS CloudHSM クラスターに接続するすべてのホストに配置します。ファイルに別の名前を付与した場合、またはホストのルート以外のパスにファイルを保存した場合には、それぞれに応じてクライアント設定ファイルを編集する必要があります。作成したばかりの証明書およびプライベートキーを使用して、次のステップでクラスター証明書署名リクエスト (CSR) に署名します。

自己署名ルート CA を使用してクラスター CSR に署名する

本稼働のクラスター用のプライベートキーの作成に使用する信頼できるハードウェアも、このキーを使用して CSR に署名するツールを提供している必要があります。次の例では、OpenSSL を使用してクラスターの CSR に署名します。以下のコマンド例では、自己署名を使用して CSR に署名します。 customerRootCA.crt

```
$ openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
  -CA <customerRootCA>.crt \
  -CAkey <customerRootCA>.key \
  -CAcreateserial \
  -out <cluster ID>_CustomerHsmCertificate.crt
Signature ok
subject=/C=US/ST=CA/O=Cavium/OU=N3FIPS/L=SanJose/CN=HSM:<HSM
 identifier>:PARTN:<partition number>, for FIPS mode
Getting CA Private Key
Enter pass phrase for <customerRootCA>.key:
```

完了すると、このコマンドは *<cluster ID>*_CustomerHsmCertificate.crt という名前のファイルを作成します。クラスターを初期化する際は、これを署名済み証明書として使用します。

署名付き証明書をルート CA と照合します (オプション) 。

```
$ openssl verify -purpose sslserver -CAfile customerRootCA.crt <cluster
 ID>_CustomerHsmCertificate.crt
<cluster ID>_CustomerHsmCertificate.crt: OK
```

自己署名ルート CA を使用して署名付き HSM 証明書を生成したら、「」に進みますステップ 4. クラスターの初期化。

中間認証機関 (ICA) チェーンを作成する

次の例では、ルート認証局 (CA) と中間 CA で構成される長さ 2 の証明書チェーンを作成する手順を説明します。最初に自己署名ルート CA 証明書を作成し、ルート CA によって署名された中間 CA を生成します。最後に、中間 CA を使用してクラスターの CSR に署名し、HSM 証明書からルート CA への完全な信頼チェーンを作成します。このアプローチは、day-to-day証明書オペレーションに中間 CA を使用しながら、ルート CA をオフラインに保つことで、セキュリティを強化します。

> **Important**
>
> 証明書チェーンを使用してクラスターを初期化するには、チェーンが次の要件を満たしている必要があります。
>
> - チェーンは、クラスター CSR に署名する中間 CA から順に並べる必要があります。この順序では、最初の ICA には、チェーン内の次の ICA の件名に一致する発行者が必要です。
> - ルート CA のみが自己署名である必要があります。つまり、発行者とサブジェクトは同一である必要があります。
> - チェーンは 4 つ以下の証明書 (末尾のルート CA を含む) で構成されている必要があり、チェーンの合計サイズは 16 KB (キロバイト) を超えることはできません。
> - すべての認証機関 (CAs) は、RFC 5280 ガイドラインに準拠する必要があります。

このセクションでは、ローカル証明書生成用の OpenSSL とマネージド証明書サービス用の AWS Private Certificate Authority (PCA) の 2 つの異なるアプローチを使用して中間認証機関チェーンを作成する例を示します。環境とセキュリティ要件に最適なアプローチを選択します。

> **Note**
>
> 次の例は一般的なユースケースであり、どちらも最も基本的な設定を使用して簡略化されています。本番環境では、ユースケースに固有の追加の設定オプションとセキュリティ要件を確認してください。

OpenSSL

CA の一般的な v3 拡張機能を使用して OpenSSL 設定ファイルを作成します。

```
$ cat > ca-extensions.conf <<EOF
[req]
distinguished_name = req_distinguished_name
[req_distinguished_name]
C = Country Name (2 letter code)
ST = State or Province Name (full name)
L = Locality Name (eg, city)
O = Organization Name (eg, company)
OU = Organizational Unit Name (eg, section)
CN = Common Name (e.g. server FQDN or YOUR name)
[v3_ca]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical,CA:true
keyUsage = critical, keyCertSign, cRLSign, digitalSignature
EOF
```

OpenSSL を使用して自己署名ルート CA を生成します。

```
$ openssl req -new -x509 -days 3652 -key customerRootCA.key -out customerRootCA.crt
 -extensions v3_ca -config ca-extensions.conf
Enter pass phrase for customerRootCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
```

```
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

中間 CA キーを生成します。

```
$ openssl genrsa -aes256 -out intermediateCA.key 2048
Generating RSA private key, 2048 bit long modulus
........+++
............+++
e is 65537 (0x10001)
Enter pass phrase for intermediateCA.key:
Verifying - Enter pass phrase for intermediateCA.key:
```

中間 CA 証明書署名リクエスト (CSR) を作成します。

```
$ openssl req -new -key intermediateCA.key -out intermediateCA.csr
Enter pass phrase for intermediateCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

自己署名ルート CA を使用して、中間 CA 証明書を作成します。

```
$ openssl x509 -req -in intermediateCA.csr \
  -CA customerRootCA.crt \
```

```
  -CAkey customerRootCA.key \
  -CAcreateserial \
  -days 3652 \
  -extensions v3_ca \
  -extfile ca-extensions.conf \
  -out intermediateCA.crt

Certificate request self-signature ok
subject=C= , ST= , L= , O= , OU=
```

証明書をチェーンファイルに結合します。

```
$ cat intermediateCA.crt customerRootCA.crt > chainCA.crt

-----BEGIN CERTIFICATE-----
[Intermediate CA]
-----END CERTIFICATE-----
...
...
-----BEGIN CERTIFICATE-----
[Root CA]
-----END CERTIFICATE-----
```

中間 CA を使用してクラスター CSR に署名します。

```
$ openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
   -CA intermediateCA.crt \
   -CAkey intermediateCA.key \
   -CAcreateserial \
   -out <cluster ID>_CustomerHsmCertificate.crt
Signature ok
subject=/C=US/ST=CA/O=Cavium/OU=N3FIPS/L=SanJose/CN=HSM:<HSM
 identifier>:PARTN:<partition number>, for FIPS mode
Getting CA Private Key
Enter pass phrase for intermediateCA.key:
```

AWS PCA

AWS Private Certificate Authority を使用してルート CA を作成してアクティブ化します。

```
$ # 1. Create Root CA
```

```
aws acm-pca create-certificate-authority \
    --certificate-authority-configuration \
        "KeyAlgorithm=RSA_4096,
        SigningAlgorithm=SHA256WITHRSA,

 Subject={Country=US,Organization=MyOrg,OrganizationalUnit=IT,CommonName=RootCA}" \
    --certificate-authority-type ROOT

# Store the Root CA Authority ARN from the previous output
ROOT_CA_AUTHORITY_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-
authority/<ca-authority-id>"

# 2. Generate Root CA CSR
aws acm-pca get-certificate-authority-csr \
    --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
    --output text > customerRootCA.csr

# 3. Self-sign Root CA Certificate
aws acm-pca issue-certificate \
    --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
    --csr fileb://customerRootCA.csr \
    --signing-algorithm SHA256WITHRSA \
 --template-arn arn:aws:acm-pca:::template/RootCACertificate/V1 \
    --validity Value=3652,Type=DAYS

# Store the Root CA certificate ARN from the previous output
ROOT_CA_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-authority/<ca-
authority-id>/certificate/<cert-id>"

# 4. Retrieve the Root CA certificate
aws acm-pca get-certificate \
    --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
    --certificate-arn $ROOT_CA_ARN \
    --output text > customerRootCA.crt

# 5. Import the Root CA Certificate
aws acm-pca import-certificate-authority-certificate \
    --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
    --certificate fileb://customerRootCA.crt
```

下位 CA (中間 CA とも呼ばれます) を作成してアクティブ化します。

```
$ # 6. Create Subordinate CA
aws acm-pca create-certificate-authority \
    --certificate-authority-configuration \
        "KeyAlgorithm=RSA_4096,
        SigningAlgorithm=SHA256WITHRSA,

 Subject={Country=US,Organization=MyOrg,OrganizationalUnit=IT,CommonName=SubordinateCA}"
 \
    --certificate-authority-type SUBORDINATE

# Store the Subordinate CA Authority ARN from the previous output
SUB_CA_AUTHORITY_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-
authority/<sub-ca-authority-id>"

# 7. Generate Subordinate CA CSR
aws acm-pca get-certificate-authority-csr \
    --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
    --output text > intermediateCA.csr

# 8. Issue Subordinate CA Certificate using Root CA
aws acm-pca issue-certificate \
    --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
    --csr fileb://intermediateCA.csr \
    --signing-algorithm SHA256WITHRSA \
    --template-arn arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1 \
    --validity Value=3651,Type=DAYS

# Store the Subordinate CA certificate ARN from the previous output
SUB_CA_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-authority/<sub-ca-
authority-id>"

# 9. Retrieve Subordinate CA Certificate
aws acm-pca get-certificate \
    --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
    --certificate-arn $SUB_CA_ARN \
    --query 'Certificate' \
    --output text > intermediateCA.crt

# 10. Import the Subordinate CA Certificate
aws acm-pca import-certificate-authority-certificate \
    --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
    --certificate fileb://intermediateCA.crt \
    --certificate-chain fileb://customerRootCA.crt
```

証明書をチェーンファイルに結合します。

```
$ cat intermediateCA.crt customerRootCA.crt > chainCA.crt

-----BEGIN CERTIFICATE-----
[Intermediate CA]
-----END CERTIFICATE-----
...
...
-----BEGIN CERTIFICATE-----
[Root CA]
-----END CERTIFICATE-----
```

AWS PCA を使用してクラスター CSR に署名します。

```
$ aws acm-pca issue-certificate \
    --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
    --csr fileb://<cluster ID>_ClusterCsr.csr \
    --signing-algorithm SHA256WITHRSA \
    --template-arn arn:aws:acm-pca:::template/EndEntityCertificate/V1 \
    --validity Value=3650,Type=DAYS

# Store your cluster's cert ARN from the previous output
CLUSTER_CERT_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-
authority/<cluster-cert-arn>"
```

署名付きクラスター証明書をダウンロードします。

```
$ aws acm-pca get-certificate \
    --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
    --certificate-arn $CLUSTER_CERT_ARN \
    --output text --query Certificate > <cluster ID>_CustomerHsmCertificate.crt
```

完了すると、このコマンドは *<cluster ID>*_CustomerHsmCertificate.crt という名前のファイルを作成します。クラスターを初期化する際は、これを署名済み証明書として使用します。

署名付き証明書を証明書チェーンと照合します (オプション) 。

```
$ openssl verify -purpose sslserver -CAfile chainCA.crt <cluster
 ID>_CustomerHsmCertificate.crt
<cluster ID>_CustomerHsmCertificate.crt: OK
```

中間 CA を使用して署名付き HSM 証明書を生成したら、 に移動しますステップ 4. クラスターの初期化。

ステップ 4. クラスターの初期化

署名済み HSM 証明書と署名証明書を使用して、クラスターを初期化します。AWS CloudHSM コンソール、AWS CLI、または AWS CloudHSM API を使用できます。

Console

クラスターを初期化するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. 検証する HSM のクラスター ID の横にあるラジオボタンをオンにします。
3. アクション を選択します。ドロップダウンメニューから 初期化 を選択します。
4. HSM を作成する 前のステップ を完了していない場合は、作成する HSM のアベイラビリティーゾーン (AZ) を選択します。次に、作成 を選択します。
5. [Download certificate signing request] ページで、[Next] を選択します。[Next] が利用できない場合は、最初にいずれかの CSR または証明書のリンクを選択します。次いで、[次へ] を選択します。
6. [Sign certificate signing request (CSR)] ページで、[Next] を選択します。
7. [Upload the certificates] ページで、以下の作業を行います。

    a. [Cluster certificate (クラスター証明書)] の横にある [Upload file (ファイルのアップロード)] を選択します。先に署名した HSM 証明書を探し選択します。前のセクションのステップを完了したら、*<cluster ID>*_CustomerHsmCertificate.crt という名前のファイルを選択します。

    b. [証明書の発行] の横にある [ファイルのアップロード] を選択します。次に、選択したアプローチに基づいて署名証明書を選択します。

    - オプション A (単一の自己署名証明書) を選択した場合: という名前のファイルを選択します。 *<customerRootCA>*.crt

- オプション B (証明書チェーン): という名前のファイルを選択します。 *<chainCA>*.crt

c. [Upload and initialize] を選択します。

AWS CLI

クラスターを初期化するには (AWS CLI)

- コマンドラインプロンプトで、initialize-cluster コマンドを実行します。以下を指定します。
  - 前に作成したクラスターの ID。
  - 前に署名した HSM 証明書。前のセクションのステップを完了すると、*<cluster ID>*_CustomerHsmCertificate.crt というファイル名で保存されています。
  - 選択した方法に基づいて証明書に署名します。
    - オプション A (単一の自己署名証明書) を選択した場合: という名前のファイルを使用します。 *<customerRootCA>*.crt
    - オプション B (証明書チェーン): という名前のファイルを使用します。 *<chainCA>*.crt

```
$ aws cloudhsmv2 initialize-cluster --cluster-id <cluster ID> \
                                    --signed-cert file://<cluster
 ID>_CustomerHsmCertificate.crt \
                                    --trust-anchor file://<customerRootCA.crt OR
 chainCA.crt>
{
    "State": "INITIALIZE_IN_PROGRESS",
    "StateMessage": "Cluster is initializing. State will change to INITIALIZED
 upon completion."
}
```

AWS CloudHSM API

クラスターを初期化するには (AWS CloudHSM API)

- 以下を使用して InitializeCluster リクエストを送信します。
  - 前に作成したクラスターの ID。

- 前に署名した HSM 証明書。前のセクションのステップを完了すると、*<cluster ID>*_CustomerHsmCertificate.crt というファイル名で保存されています。
- 選択した方法に基づいて証明書に署名します。
  - オプション A (単一の自己署名証明書) を選択した場合: という名前のファイルを使用します。 *<customerRootCA>*.crt
  - オプション B (証明書チェーン): という名前のファイルを使用します。 *<chainCA>*.crt

CloudHSM CLI をインストールして設定する

AWS CloudHSM クラスター内の HSM を操作するには、CloudHSM CLI が必要です。

クライアントインスタンスに接続し、次のコマンドを実行して、 AWS CloudHSM コマンドラインツールをダウンロードしてインストールします。詳細については、「とやり取りするための Amazon EC2 クライアントインスタンスを起動する AWS CloudHSM」を参照してください。

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

ARM64 アーキテクチャの Amazon Linux 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-
latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

ARM64 の Amazon Linux 2 アーキテクチャ:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

x86_64 アーキテクチャの RHEL 10:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.x86_64.rpm
```

ARM64 アーキテクチャの RHEL 10:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

ARM64 アーキテクチャの RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

ARM64 アーキテクチャの RHEL 8:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.aarch64.rpm
```

Ubuntu 26.04 LTS

x86_64 アーキテクチャの Ubuntu 26.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/cloudhsm-cli_latest_u26.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u26.04_amd64.deb
```

ARM64 アーキテクチャの Ubuntu 26.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/cloudhsm-cli_latest_u26.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u26.04_arm64.deb
```

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-cli_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_amd64.deb
```

ARM64 アーキテクチャの Ubuntu 24.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-cli_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

ARM64 アーキテクチャの Ubuntu 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 の場合、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 の場合、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 の場合、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

次のコマンドを使用して CloudHSM CLI を設定します。

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the
 HSMs>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI
 IPv4 / IPv6 addresses of the HSMs>
```

でクラスターをアクティブ化する AWS CloudHSM

AWS CloudHSM クラスターをアクティブ化すると、クラスターの状態が初期化からアクティブに変わります。その後、ハードウェアセキュリティモジュール (HSM) のユーザーを管理 し、HSM を使用します。

Important

クラスターをアクティブ化する前に、クラスターに接続する各 EC2 インスタンス上のプラットフォームのために、デフォルトの場所に発行証明書をコピーする必要があります (クラスターを初期化するときに、発行証明書を作成します)。クラスターの初期化時に選択したアプローチに基づいて、適切な証明書ファイルを使用します。

- オプション A (単一の自己署名証明書): コピー `customerRootCA.crt`
- オプション B (証明書チェーン): コピー `chainCA.crt`

Linux の場所:

```
/opt/cloudhsm/etc/<customerRootCA.crt OR chainCA.crt>
```

Windows の場所:

```
C:\ProgramData\Amazon\CloudHSM\<customerRootCA.crt OR chainCA.crt>
```

証明書ファイルをコピーしたら、`/opt/cloudhsm/etc/cloudhsm-cli.cfg`ファイルを編集して、証明書ファイル名がコピーした CA 証明書の名前と一致することを確認します。

発行証明書を配置したら、CloudHSM CLI をインストールし、最初の HSM で cluster activate コマンドを実行します。クラスター内の最初の HSM の管理者アカウントに unactivated-admin ロールが割り当てられているはずです。これはクラスターがアクティブ化される前にのみ存在する一時的なロールです。クラスターをアクティブ化すると、非アクティブ化された管理者のロールは管理者に変わります。

クラスターをアクティブ化するには

1. 以前に起動したクライアントインスタンスに接続します。詳細については、「とやり取りするための Amazon EC2 クライアントインスタンスを起動する AWS CloudHSM」を参照してください。Linux インスタンスまたは Windows Server を起動できます。
2. CloudHSM CLI をインタラクティブモードで実行します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

3. (オプション) user list コマンドを使用して、既存のユーザーを表示します。

   ```
   aws-cloudhsm > user list
   {
     "error_code": 0,
     "data": {
       "users": [
         {
           "username": "admin",
           "role": "unactivated-admin",
           "locked": "false",
           "mfa": [],
           "cluster-coverage": "full"
         },
         {
           "username": "app_user",
           "role": "internal(APPLIANCE_USER)",
           "locked": "false",
           "mfa": [],
           "cluster-coverage": "full"
         }
       ]
     }
   }
   ```

4. cluster activate コマンドを使用して初期管理者パスワードを設定します。

```
aws-cloudhsm > cluster activate
Enter
 password:<NewPassword>
Confirm password:<NewPassword>
{
  "error_code": 0,
  "data": "Cluster activation successful"
}
```

新しいパスワードをパスワードワークシートに書き留めておくことをお勧めします。ワークシートを紛失しないでください。パスワードワークシートのコピーを印刷することをお勧めします。このコピーに重要な HSM のパスワードをメモし、安全な場所に保存してください。また、このワークシートのコピーをオフサイトの安全なストレージに保存することをお勧めします。

5. (オプション) user list コマンドを使用して、ユーザーのタイプが admin/CO に変更されていることを確認します。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
       {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

6. quit コマンドを使用して、CloudHSM CLI ツールを停止します。

```
aws-cloudhsm > quit
```

CMU または CloudHSM CLI のオペレーションの詳細については、HSM ユーザーについて と CMU での HSM ユーザー管理について を参照してください。

クライアントと の間の相互 TLS を設定する AWS CloudHSM (推奨)

以下のトピックでは、クライアントと 間の相互 Transport Layer Security (mTLS) を有効にするために完了する必要があるステップについて説明します AWS CloudHSM。

考慮事項

- 現在、この機能は hsm2m.medium でのみ利用できます。HSM タイプの詳細については、「AWS CloudHSM クラスターモード」を参照してください。
- で使用する AWS CloudHSM キーストアでは、mTLS はサポートされていません AWS Key Management Service。

トピック

- ステップ 1. トラストアンカーを作成して HSM に登録する
- ステップ 2. の mTLS を有効にする AWS CloudHSM
- ステップ 3. の mTLS 適用を設定する AWS CloudHSM

ステップ 1. トラストアンカーを作成して HSM に登録する

mTLS を有効にする前に、トラストアンカーを作成して HSM に登録する必要があります。これは 2 ステップのプロセスです。

トピック

- プライベートキーを使用して、自己署名ルート証明書を作成します。
- トラストアンカーを HSM に登録する

プライベートキーを使用して、自己署名ルート証明書を作成します。

> **Note**
>
> 実稼働用クラスターでは、作成しようとしているキーはランダム性の信頼できるソースを使用して安全な方法で作成されている必要があります。安全なオフサイトあるいはオフライン HSM またはその同等を使用することが推奨されます。キーを安全に保存します。
> 開発とテストでは、任意の便利なツール (OpenSSL など) を使用してキーを作成し、ルート証明書に自己署名できます。mTLS を有効にする AWS CloudHSM でクライアント証明書に署名するには、 キーとルート証明書が必要です。

次の例は、OpenSSL を使用してプライベートキーと自己署名ルート証明書を作成する方法を示しています。

Example— OpenSSL でプライベートキーを作成する

次のコマンドを使用して、AES-256 アルゴリズムで暗号化された 4096 ビット RSA キーを作成します。この例を使用するには、*<mtls_ca_root_1.key>* を、キーの保存先のファイル名に置き換えてください。

```
$ openssl genrsa -out <mtls_ca_root_1.key> -aes256 4096
Generating RSA private key, 4096 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
Enter pass phrase for mtls_ca_root_1.key:
Verifying - Enter pass phrase for mtls_ca_root_1.key:
```

Example– OpenSSL を使用して自己署名ルート証明書を作成する

次のコマンドを使用して、作成したプライベートキーから mtls_ca_root_1.crtという名前の自己署名ルート証明書を作成します。証明書の有効期間は 25 年間 (9130 日) です。画面の指示を読み、プロンプトに従います。

```
$ openssl req -new -x509 -days 9130 -key mtls_ca_root_1.key -out mtls_ca_root_1.crt
Enter pass phrase for mtls_ca_root_1.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

トラストアンカーを HSM に登録する

自己署名ルート証明書を作成した後、管理者はそれを信頼アンカーとして AWS CloudHSM クラスターに登録する必要があります。

HSM にトラストアンカーを登録するには

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、管理者としてログインします。

   ```
   aws-cloudhsm > login --username <admin> --role admin
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "<admin>",
       "role": "admin"
     }
   }
   ```

3. CloudHSM CLI でトラストアンカーを登録する コマンドを使用してトラストアンカーを登録します。詳細については、次の例を参照するか、または help cluster mtls register-trust-anchor コマンドを使用してください。

Example— AWS CloudHSM クラスターにトラストアンカーを登録する

以下の例では、CloudHSM CLI で cluster mtls register-trust-anchor コマンドを使用して、トラストアンカーを HSM に登録する方法を示しています。このコマンドを使用するには、管理者が HSM にログインしている必要があります。以下の値を自分の値に置き換えてください。

```
aws-cloudhsm > cluster mtls register-trust-anchor --path </path/mtls_ca_root_1.crt>
{
  "error_code": 0,
  "data": {
    "trust_anchor": {
      "certificate-reference": "0x01",
      "certificate": "<PEM Encoded Certificate>",
      "cluster-coverage": "full"
    }
  }
}
```

> **Note**
>
> AWS CloudHSM では、中間証明書をトラストアンカーとして登録できます。このような場合は、PEM でエンコードされた証明書チェーンファイル全体を、証明書を階層順にして、HSM に登録する必要があります。
> AWS CloudHSM は、6980 バイトの証明書チェーンをサポートします。

トラストアンカーを正常に登録したら、cluster mtls list-trust-anchors コマンドを実行して、以下に示すように、現在登録されているトラストアンカーを確認できます。

```
aws-cloudhsm > cluster mtls list-trust-anchors
{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
```

```
        "certificate": "<PEM Encoded Certificate>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```

> **Note**
>
> hsm2m.medium に登録できるトラストアンカーの最大数は 2 です。

ステップ 2. の mTLS を有効にする AWS CloudHSM

の mTLS を有効にするには AWS CloudHSM、「Create and register a trust anchor on the HSM」で生成したルート証明書によって署名されたプライベートキーとクライアント証明書を作成し、任意の Client SDK 5 設定ツールを使用してプライベートキーパスとクライアント証明書チェーンパスを設定する必要があります。

トピック

- プライベートキーとクライアント証明書チェーンを作成する
- クライアント SDK 5 の mTLS を設定する

プライベートキーとクライアント証明書チェーンを作成する

Example— OpenSSL でプライベートキーを作成する

次のコマンドを使用して、4,096 ビット RSA キーを作成します。この例を使用するには、*<ssl-client.key>* を、キーの保存先のファイル名に置き換えてください。

```
$ openssl genrsa -out <ssl-client.key> 4096
Generating RSA private key, 4096 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
```

Example– OpenSSL を使用して証明書署名リクエスト (CSR) を生成する

次のコマンドを使用して、作成したプライベートキーから証明書署名リクエスト (CSR) を生成します。画面の指示を読み、プロンプトに従います。

```
$ openssl req -new -key <ssl-client.key> -out <ssl-client.csr>
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Example– ルート証明書を使用して CSR に署名する

次のコマンドを使用して、[トラストアンカーを作成して HSM に登録する] で作成および登録したルート証明書を使用して CSR に署名し、ssl-client.crt という名前のクライアント証明書を作成します。証明書の有効期間は 5 年間 (1826 日) です。

```
$ openssl x509 -req -days 1826 -in <ssl-client.csr> -CA <mtls_ca_root_1.crt> -
CAkey <mtls_ca_root_1.key> -CAcreateserial -out <ssl-client.crt>
```

Example– クライアント証明書チェーンを作成する

次のコマンドを使用して、[トラストアンカーを作成して HSM に登録する] で作成および登録したクライアント証明書とルート証明書を組み合わせて、ssl-client.pem という名前のクライアント証明書チェーンを作成します。これは次の手順での設定に使用されます。

```
$ cat <ssl-client.crt> <mtls_ca_root_1.crt> > <ssl-client.pem>
```

> **Note**
>
> [トラストアンカーを作成して HSM に登録する] で中間証明書をトラストアンカーとして登録した場合は、必ずクライアント証明書と証明書チェーン全体を組み合わせてクライアント証明書チェーンを作成してください。

クライアント SDK 5 の mTLS を設定する

任意のクライアント SDK 5 設定ツールを使用して、適切なクライアントキーパスとクライアント証明書チェーンパスを指定することで、相互 TLS を有効にします。クライアント SDK 5 の設定ツールの詳細については、「???」 を参照してください。

PKCS #11 library

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
$ sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで `ssl-client.pem`、`ssl-client.key` を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して `ssl-client.pem` と `ssl-client.key` を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

OpenSSL Dynamic Engine

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで `ssl-client.pem`、`ssl-client.key` を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Key Storage Provider (KSP)

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して `ssl-client.pem` と `ssl-client.key` を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
```

```
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

JCE provider

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで ssl-client.pem、ssl-client.key を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して ssl-client.pem と ssl-client.key を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

CloudHSM CLI

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで `ssl-client.pem`、`ssl-client.key` を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して `ssl-client.pem` と `ssl-client.key` を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

ステップ 3. の mTLS 適用を設定する AWS CloudHSM

クライアント SDK 5 の設定ツールを使用して を設定した後、クライアントと 間の接続 AWS CloudHSM はクラスター内の相互 TLS になります。ただし、設定ファイルからプライベートキーパスとクライアント証明書チェーンパスを削除すると、接続が再び通常の TLS になります。CloudHSM CLI を使用して、次の手順を実行してクラスター内の mtls 強制を設定できます。

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、管理者としてログインします。

   ```
   aws-cloudhsm > login --username <admin> --role admin
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "<admin>",
       "role": "admin"
     }
   }
   ```

   > **Note**
   >
   > 1. CloudHSM CLI が設定されていることを確認し、mTLS 接続で CloudHSM CLI を起動します。
   > 2. mTLS 適用を設定する前に、admin としてのユーザー名を持つ管理者が HSM にログインする必要があります。

3. CloudHSM CLI で mTLS 適用レベルを設定する コマンドを使用して適用を設定します。詳細については、次の例を参照するか、または help cluster mtls set-enforcement コマンドを使用してください。

   Example– AWS CloudHSM クラスターで mTLS 強制を設定する

   次の例は、CloudHSM CLI で cluster mtls set-enforcement コマンドを使用して HSM で mTLS 適用を設定する方法を示しています。このコマンドを使用するには、ユーザー名が admin の管理者が HSM にログインする必要があります。

   ```
   aws-cloudhsm > cluster mtls set-enforcement --level cluster
   ```

```
{
  "error_code": 0,
  "data": {
    "message": "Mtls enforcement level set to Cluster successfully"
  }
}
```

> **Warning**
>
> クラスターで mTLS の使用を適用すると、既存の非 mTLS 接続はすべて削除され、mTLS 証明書を持つクラスターにのみ接続できます。

でキーを作成して使用する AWS CloudHSM

新しいクラスターでキーを作成して使用する前に、 AWS CloudHSM CLI を使用してハードウェアセキュリティモジュール (HSM) ユーザーを作成します。詳細については、「HSM ユーザー管理タスクを理解する」、AWS CloudHSM コマンドラインインターフェイス (CLI) の開始方法」、および「HSM ユーザーを管理する方法」を参照してください。

> **Note**
>
> クライアント SDK 3 を使用している場合は、CloudHSM CLI の代わりに CloudHSM 管理ユーティリティ (CMU) を使用してください。

HSM ユーザーを作成したら、HSM にサインインし、次のいずれかのオプションを使用してキーを管理できます。

- key management utility、コマンドラインツール を使用
- PKCS #11 library を使用して C アプリケーションを構築する
- JCE provider を使用して Java アプリケーションを構築する
- OpenSSL Dynamic Engine directly from the command line を使用
- NGINX and Apache web servers を用いて、TLS オフロードのために [OpenSSL Dynamic Engine] を使用
- Microsoft Windows Server 認証機関 (CA) AWS CloudHSM で のキーストレージプロバイダー (KSP) を使用する ???

- Microsoft Sign Tool AWS CloudHSM で のキーストレージプロバイダー (KSP) を使用する
- Internet Information Server (IIS) Web サーバーの TLS オフロードでキーストレージプロバイダー (KSP) を使用する

のベストプラクティス AWS CloudHSM

AWS CloudHSMを効果的に使用するには、このトピックのベストプラクティスを実行してください。

内容

- AWS CloudHSM クラスター管理のベストプラクティス
- AWS CloudHSM ユーザー管理のベストプラクティス
- AWS CloudHSM キー管理のベストプラクティス
- AWS CloudHSM アプリケーション統合のベストプラクティス
- AWS CloudHSM モニタリングのベストプラクティス

AWS CloudHSM クラスター管理のベストプラクティス

AWS CloudHSM クラスターを作成、アクセス、管理するときは、このセクションのベストプラクティスに従ってください。

クラスターをスケールしてピークトラフィックを処理する

クラスターが処理できる最大スループットには、クライアントインスタンスのサイズ、クラスターサイズ、ネットワークトポグラフィー、ユースケースに必要な暗号化オペレーションなど、いくつかの要因が影響する可能性があります。

手始めに、一般的なクラスターのサイズと構成に関するパフォーマンスの見積もりに関するトピック AWS CloudHSM パフォーマンス情報 を参照してください。現在のアーキテクチャが耐障害性に優れていて、適切な規模になっているかを判断するために、予想されるピーク負荷でクラスターの負荷テストを行うことを推奨します。

高可用性対応のクラスターをアーキテクチャー

メンテナンスを考慮して冗長性を追加します。スケジュールされたメンテナンスや問題を検出した場合は、HSM を AWS 置き換えることができます。原則として、クラスターサイズには少なくとも +1 の冗長性が必要です。たとえば、ピーク時にサービスを稼働させるために HSM が 2 つ必要であれば、理想的なクラスターサイズは 3 つになります。可用性に関するベストプラクティスに従えば、HSM を置き換えてもサービスに影響はないはずです。ただし、交換した HSM で進行中のオペレーションは失敗する可能性があるため、再試行する必要があります。

HSM を複数のアベイラビリティーゾーンに分散させる: アベイラビリティーゾーンが停止した際に、サービスをどのように運用できるかを検討してください。 AWS では、HSM をできるだけ多くのアベイラビリティーゾーンに分散することを推奨しています。HSM が 3 つあるクラスターでは、HSM を 3 つのアベイラビリティーゾーンに分散する必要があります。システムによっては、追加の冗長性が必要な場合があります。

新しく生成されたキーの耐久性を確保するために、HSM を少なくとも 3 つ用意してください

新しく生成されたキーの耐久性が必要なアプリケーションの場合は、リージョン内の異なるアベイラビリティーゾーンに少なくとも 3 つの HSM を分散させることを推奨します。

クラスターへの安全なアクセス

プライベートサブネットを使用してインスタンスへのアクセスを制限する: HSM とクライアントインスタンスは、VPC のプライベートサブネットで起動します。これにより、外部からの HSM へのアクセスが制限されます。

VPC エンドポイントを使用して APIs にアクセスする: AWS CloudHSM データプレーンは、インターネットや AWS APIs にアクセスすることなく動作するように設計されています。クライアントインスタンスが AWS CloudHSM API にアクセスする必要がある場合は、クライアントインスタンスでインターネットアクセスを必要とせずに、VPC エンドポイントを使用して API にアクセスできます。詳細については「AWS CloudHSM および VPC エンドポイント」を参照してください。

ニーズに合わせてスケールすることでコストを削減

AWS CloudHSMの使用には初期費用はかかりません。HSM を終了するまで、HSM を起動するたびに 1 時間あたりの料金をお支払いいただきます。サービスで の継続的な使用が必要ない場合は AWS CloudHSM、HSMsが不要になったときにゼロにスケールダウン (削除) することでコストを削減できます。HSM が再び必要になった場合は、HSM をバックアップから復元できます。たとえば、月に一度、具体的にはその月の最終日にコードに署名する必要があるワークロードがある場合は、その前にクラスターをスケールアップし、作業完了後に HSM を削除してスケールダウンし、翌月末にクラスターを復元して署名オペレーションを再実行することができます。

AWS CloudHSM は、クラスター内の HSMsの定期的なバックアップを自動的に作成します。後で新しい HSM を追加すると、 AWS CloudHSM は最新のバックアップを新しい HSM に復元し、残したのと同じ場所から使用を再開できるようにします。 AWS CloudHSM アーキテクチャコストを計算するには、AWS CloudHSM 「料金表」を参照してください。

関連リソース:

- バックアップの一般的な概要
- バックアップの保持ポリシー
- AWS リージョン間での AWS CloudHSM クラスターバックアップのコピー

AWS CloudHSM ユーザー管理のベストプラクティス

AWS CloudHSM クラスター内のユーザーを効果的に管理するには、このセクションのベストプラクティスに従います。HSM ユーザーは IAM ユーザーとは異なります。適切な権限を持つ ID ベースのポリシーを持つ IAM ユーザーとエンティティは、AWS API を介してリソースを操作することで HSM を作成できます。HSM を作成したら、HSM ユーザー認証情報を使用して HSM でのオペレーションを認証する必要があります。HSM ユーザーの詳細なガイドについては、「の HSM ユーザー AWS CloudHSM」を参照してください。

HSM ユーザーの認証情報の保護

HSM ユーザーは HSM にアクセスして暗号化オペレーションや管理オペレーションを実行できるため、HSM ユーザーの認証情報を安全に保護することが不可欠です。 AWS CloudHSM は HSM ユーザー認証情報にアクセスできないため、認証情報にアクセスできなくなった場合はサポートできません。

ロックアウトを防ぐため、少なくとも 2 人の管理者を配置します

クラスターからロックアウトされないように、1 つの管理者パスワードを紛失した場合に備えて、少なくとも 2 人の管理者を配置することをお勧めします。このような場合は、もう一方の管理者にパスワードをリセットしてもらうことができます。

> **Note**
>
> クライアント SDK 5 の管理者は、クライアント SDK 3 の Crypto Officer (CO) と同義です。

すべてのユーザー管理オペレーションでクォーラムを有効にします

クォーラムでは、ユーザー管理オペレーションを承認しないと実行できない管理者の最小数を設定できます。管理者には権限があるため、すべてのユーザー管理オペレーションでクォーラムを有効にす

ることをお勧めします。これにより、管理者パスワードのいずれかが侵害された場合の影響を抑えることができます。詳細については、Managing Quorum を参照してください。

各自の権限を制限した複数の Crypto User を作成する

Crypto User の責任を分離することで、1 人のユーザーがシステム全体を完全に制御できなくなります。このため、複数の Crypto User を作成し、それぞれの権限を制限するようお勧めします。通常、これは異なる Crypto User にそれぞれ異なる責任と実行するアクションを明確に与えることによって行われます (たとえば、1 人の暗号ユーザーがキーを生成して他の暗号ユーザーと共有し、その暗号ユーザーをアプリケーションで利用させるなど) 。

関連リソース:

- CloudHSM CLI を使用してキーを共有する
- CloudHSM CLI を使用してキーの共有を解除する

AWS CloudHSM キー管理のベストプラクティス

AWS CloudHSMでキーを管理する場合は、このセクションのベストプラクティスに従います。

適切なキーのタイプを選択する

セッションキーを使用する場合、1 秒あたりのトランザクション数 (TPS) は、そのキーが存在する 1 つの HSM に制限されます。クラスターに HSM を追加しても、そのキーのリクエストのスループットは向上しません。同じアプリケーションにトークンキーを使用すると、リクエストはクラスター内の利用可能なすべての HSM に負荷分散されます。詳細については、「でのキー同期と耐久性の設定 AWS CloudHSM」を参照してください。

キーのストレージ制限を管理する

HSM には、HSM に一度に保存できるトークンとセッションキーの最大数に制限があります。キーストレージの制限に関する詳細は、「AWS CloudHSM クォータ」を参照してください。アプリケーションが制限を超えるものを必要とする場合は、次の 1 つまたは複数の方法を使用してキーを効果的に管理できます。

トラステッドラッピングを使用してキーを外部データストアに保存する:トラステッドキーラッピングを使用すると、ラップされたすべてのキーを外部データストア内に保存することで、キースト

レージの制限を回避できます。このキーを使用する必要がある場合は、キーをセッションキーとして HSM にアンラップし、そのキーを必要なオペレーションに使用してからセッションキーを破棄できます。元のキーデータはデータストアに安全に保存され、必要なときにいつでも使用できます。そのために信頼できるキーを使用することで、最大限の保護が可能になります。

キーをクラスターに分散する: キーストレージの制限を克服するためのもう 1 つの方法は、キーを複数のクラスターに保存することです。このアプローチでは、各クラスターに保存されているキーのマッピングを維持します。このマッピングを使用して、必要なキーを含むクライアントリクエストをクラスターにルーティングします。同じクライアントアプリケーションから複数のクラスターに接続する方法の詳細については、以下のトピックを参照してください。

- JCE プロバイダーを使用した複数の AWS CloudHSM クラスターへの接続
- 用の PKCS #11 ライブラリを使用した複数のスロット設定 AWS CloudHSM

キーラッピングの管理と保護

キーには、EXTRACTABLE 属性によって抽出可能または抽出不可のマークが付けられます。デフォルトでは、HSM キーは抽出可能とマークされています。

抽出可能なキーとは、キーラッピングによって HSM からエクスポートできるキーのことです。ラップされたキーは暗号化されるため、使用する前に同じラップキーを使用してラップを解除する必要があります。抽出不可能なキーは、いかなる状況においても HSM からエクスポートすることはできません。抽出不可能なキーを抽出可能にする方法はありません。このため、キーを抽出可能にする必要があるかどうかを検討し、それに応じて対応するキー属性を設定することが重要です。

アプリケーションでキーラッピングが必要な場合は、信頼できるキーラッピングを利用して、管理者によって明示的に信頼できるとマークされたキーのみを HSM ユーザーがラップ/アンラップできるように制限する必要があります。詳細については、のキー AWS CloudHSM の「信頼できるキーラッピングに関するトピック」を参照してください。

関連リソース

- ラップ関数とアンラップ関数
- JCE の暗号関数
- AWS CloudHSM クライアント SDK 5 でサポートされている Java キー属性
- CloudHSM CLI のキー属性

AWS CloudHSM アプリケーション統合のベストプラクティス

このセクションのベストプラクティスに従って、アプリケーションと AWS CloudHSM クラスターの統合方法を最適化します。

クライアント SDK のブートストラップ

クライアント SDK をクラスターに接続する前に、ブートストラップする必要があります。IP アドレスをクラスターにブートストラップするときは、可能であれば `--cluster-id` パラメーターを使用することをおすすめします。この方法では、個々のアドレスを追跡しなくても、クラスター内のすべての HSM IP アドレスが設定に入力されます。これにより、HSM がメンテナンス中であったり、アベイラビリティーゾーンが停止した場合でも、アプリケーションの初期化の回復力がさらに高まります。詳細については、クライアント SDK をブートストラップするを参照してください。

認証してオペレーションを実行

では AWS CloudHSM、暗号化オペレーションなどのほとんどのオペレーションを実行する前に、クラスターに対して認証する必要があります。

CloudHSM CLI による認証: CloudHSM CLI による認証は、シングルコマンドモード または インタラクティブモード のいずれかを使用して行うことができます。CloudHSM CLI を使用して HSM にログインする コマンドを使用してインタラクティブモードで認証します。シングルコマンドモードで認証するには、環境変数 `CLOUDHSM_ROLE` および `CLOUDHSM_PIN` を設定する必要があります。その方法の詳細については、「シングルコマンドモード」を参照してください。 AWS CloudHSM では、アプリケーションで使用されていないときは、HSM 資格情報を安全に保管することをお勧めします。

PKCS #11 による認証: PKCS #11 では、C_OpenSession を使用してセッションを開いた後に C_Login API を使用してログインします。C_Login はスロット (クラスター) ごとに 1 つだけ実行する必要があります。ログインに成功したら、追加のログインオペレーションを行わなくても C_OpenSession を使用して追加のセッションを開くことができます。PKCS #11 への認証の例については、「AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリのコードサンプル」を参照してください。

JCE による認証: AWS CloudHSM JCE プロバイダーは、暗黙的なログインと明示的なログインの両方をサポートしています。適切な方法は、ユースケースによって異なります。アプリケーションがクラスターから切断され、再認証が必要になった場合、SDK が自動的に認証を処理するため、可能な場合は Implicit Login を使用することをお勧めします。暗黙的ログインを使用すると、アプリケーションコードを制御できないインテグレーションを使用する場合でも、アプリケーションに認証情報

を提供できます。ログイン方法の詳細については、「ステップ 2: JCE プロバイダーに認証情報を提供する」を参照してください。

OpenSSL による認証: OpenSSL 動的エンジンでは、環境変数を使用して認証情報を指定します。AWS CloudHSM では、HSM 認証情報は、アプリケーションで使用しないときは安全に保存することをお勧めします。可能であれば、手動で入力しなくてもこれらの環境変数を体系的に取得して設定するように環境を設定する必要があります。OpenSSL による認証の詳細については、「AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine をインストールする」を参照してください。

KSP による認証: キーストレージプロバイダー (KSP) で認証するには、Windows 資格情報マネージャーまたは環境変数のいずれかを使用できます。AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) をインストールする を参照してください。

アプリケーションのキーを効果的に管理する

キー属性を使用してキーで実行できる操作を制御する: キーを生成するときは、キー属性を使用して、そのキーに対する特定の種類の操作を許可または拒否する一連の権限を定義します。キーは、タスクを完了するのに必要な最小限の属性で生成することをおすすめします。たとえば、暗号化に使用する AES キーを HSM からラップアウトすることも許可しないでください。詳細については、以下のクライアント SDK の属性ページを参照してください。

- PKCS #11 の主要属性
- JCE キーの属性

可能な場合は、キーオブジェクトをキャッシュしてレイテンシーを最小限に抑える: キー検索オペレーションはクラスター内のすべての HSM にクエリを実行します。このオペレーションはコストがかかり、クラスター内の HSM 数に合わせて拡張できません。

- PKCS #11 では、 `C_FindObjects` API を使用してキーを検索します。
- JCE では、KeyStore を使用してキーを検索します。

最適なパフォーマンスを得るには、アプリケーションの起動時にキー検索コマンド (KMU を使用して属性で AWS CloudHSM キーを検索するや などCloudHSM CLI でユーザーのキーを一覧表示する) を 1 回だけ使用し、返されたキーオブジェクトをアプリケーションメモリにキャッシュ AWS することをお勧めします。後でこのキーオブジェクトが必要になった場合は、オペレーションのたびにこのオブジェクトをクエリするのではなく、キャッシュからオブジェクトを取得する必要があります。そうすると、パフォーマンスのオーバーヘッドが大幅に増加します。

できるだけ少ない呼び出しでオブジェクト属性を読み取る

読み取り属性はクラスターにリクエストを送信し、リクエストの数は使用する SDK によって異なります。

- PKCS #11 ライブラリでは、テンプレートに含まれる属性の数に関係なく、各`C_GetAttributeValue`呼び出しが 1 つのリクエストを送信します。1 つのテンプレートで必要なすべての属性をリクエストします。詳細については、「AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリを使用して属性を取得する」を参照してください。
- JCE では、リクエストする各属性には 1 つのリクエストの費用がかかります。アプリケーションに必要な属性のみをリクエストします。

どちらの場合も、値をアプリケーションにキャッシュし、何かがオブジェクトを変更するまで再利用します。

マルチスレッドを使用してください

AWS CloudHSM はマルチスレッドアプリケーションをサポートしていますが、マルチスレッドアプリケーションには注意すべき点がいくつかあります。

PKCS #11 では、PKCS #11 ライブラリ (呼び出し側`C_Initialize`) は 1 回だけ初期化する必要があります。各スレッドには独自のセッションを割り当てる必要があります (`C_OpenSession`)。同じセッションを複数のスレッドで使用することはお勧めしません。

JCE では、 AWS CloudHSM プロバイダーは 1 回だけ初期化する必要があります。SPI オブジェクトのインスタンスをスレッド間で共有しないでください。たとえば、Cipher、Signature、Digest、Mac、KeyFactory、KeyGenerator の各オブジェクトは、それぞれのスレッドのコンテキストでのみ使用してください。

スロットリングエラーの処理

以下の条件下では、HSM スロットリングエラーが発生する可能性があります。

- クラスターがピーク時のトラフィックを管理できるよう適切にスケーリングされていない。
- メンテナンスイベント中は、クラスターのサイズが +1 の冗長性を持たない。
- アベイラビリティーゾーンが停止すると、クラスターで使用可能な HSM の数が減少する。

このシナリオを最適に処理する方法については、「HSM スロットリング」を参照してください。

クラスターのサイズが適切でスロットリングされないように、 では、予想されるピークトラフィックを使用して環境で負荷テストを行う AWS ことをお勧めします。

クラスターオペレーションのリトライを統合してください

AWS は、運用上またはメンテナンス上の理由で HSM を置き換える場合があります。このような状況に対してアプリケーションの耐障害性を高めるため、 AWS では、クラスターにルーティングされるすべてのオペレーションでクライアント側の再試行ロジックを実装することをお勧めします。置換によって失敗したオペレーションを後で再試行しても、成功することが期待されます。

ディザスタリカバリ戦略の実装

イベントに対応して、トラフィックをクラスター全体またはリージョン全体から遠ざける必要がある場合があります。以下のセクションでは、そのための複数の戦略について説明します。

VPC ピアリングを使用して別のアカウントまたはリージョンからクラスターにアクセスする: VPC ピアリングを使用して、別のアカウントまたはリージョンから AWS CloudHSM クラスターにアクセスできます。詳細については、VPC Peering Guideの「VPC ピア機能とは」を参照してください。ピアリング接続を確立し、セキュリティグループを適切に設定すると、通常と同じ方法で HSM IP アドレスと通信できます。

同じアプリケーションから複数のクラスターに接続: クライアント SDK 5 の JCE プロバイダー、PKCS #11 ライブラリ、および CloudHSM CLI は、同じアプリケーションから複数のクラスターへの接続をサポートします。たとえば、それぞれ異なるリージョンにある 2 つのアクティブなクラスターがある場合、アプリケーションは両方に一度に接続して、通常のオペレーションの一部としてこの 2 つのクラスター間の負荷を分散できます。アプリケーションが クライアント SDK 5 (最新の SDK) を使用していない場合、同じアプリケーションから複数のクラスターに接続することはできません。あるいは、別のクラスターを稼働させ続け、地域的な障害が発生した場合には、ダウンタイムを最小限に抑えるためにトラフィックを他のクラスターに移すこともできます。詳細については、それぞれのページを参照してください。

- 用の PKCS #11 ライブラリを使用した複数のスロット設定 AWS CloudHSM
- JCE プロバイダーを使用した複数の AWS CloudHSM クラスターへの接続
- CloudHSM CLI を使用した複数のクラスターへの接続

バックアップからのクラスターの復元: 既存のクラスターのバックアップから新しいクラスターを作成できます。詳細については、「でのクラスターバックアップ AWS CloudHSM」を参照してください。

アプリケーションのデプロイをずらす

クライアントアプリケーションのデプロイと再起動には、プログレッシブウェーブベースのデプロイ、ワンボックスデプロイ、ローリングデプロイなどのずらされたデプロイ戦略に従います。このアプローチにより、変更による潜在的な影響を最小限に抑えながら、デプロイ中に本番トラフィックを処理するのに十分な容量を確保できます。

AWS CloudHSM モニタリングのベストプラクティス

このセクションでは、クラスターとアプリケーションのモニタリングに使用できる複数のメカニズムについて説明します。モニタリングの詳細については、「モニタリング AWS CloudHSM」を参照してください。

クライアントログのモニタリング

すべてのクライアント SDK には、監視可能なログが書き込まれます。ログ記録の詳細については、「AWS CloudHSM クライアント SDK ログの使用」を参照してください。

Amazon ECS や などの一時的なプラットフォームでは AWS Lambda、ファイルからクライアントログを収集することは難しい場合があります。このような状況では、ログをコンソールに書き込むようにクライアント SDK ロギングを設定するのがベストプラクティスです。ほとんどのサービスでは、この出力を自動的に収集して Amazon CloudWatch Logsに公開し、ユーザーが保存して確認できるようにします。

AWS CloudHSM クライアント SDK 上でサードパーティー統合を使用している場合は、出力をコンソールにもログ記録するようにソフトウェアパッケージを設定する必要があります。 AWS CloudHSM クライアント SDK からの出力は、このパッケージによってキャプチャされ、それ以外の場合は独自のログファイルに書き込まれる場合があります。

アプリケーションのロギングオプションの設定方法については、「AWS CloudHSM クライアント SDK 5 設定ツール」を参照してください。

監査ログのモニタリング

AWS CloudHSM はAmazon CloudWatch アカウントに監査ログを発行します。監査ログは HSM から取得され、監査目的で特定のオペレーションを追跡します。

監査ログを使用して、HSM で呼び出された管理コマンドを追跡できます。たとえば、予期しない管理オペレーションが実行されていることに気付いたときにアラームをトリガーできます。

詳細については、「HSM 監査ログの記録の仕組み」を参照してください。

モニタリング AWS CloudTrail

AWS CloudHSM は と統合されています。これは AWS CloudTrail、ユーザー、ロール、または のサービスによって実行されたアクションを記録する AWS サービスです AWS CloudHSM。 は、 のすべての API コールをイベント AWS CloudHSM として AWS CloudTrail キャプチャします。キャプチャされた呼び出しには、 AWS CloudHSM コンソールからの呼び出しと AWS CloudHSM API オペレーションへのコード呼び出しが含まれます。

AWS CloudTrail を使用して、 AWS CloudHSM コントロールプレーンに対して行われた API コールを監査し、アカウントで望ましくないアクティビティが発生していないことを確認できます。

詳細については、「AWS CloudTrailおよび の使用AWS CloudHSM」を参照してください。

Amazon CloudWatch メトリクスのモニタリング

Amazon CloudWatch メトリクスを使用して、 AWS CloudHSM クラスターをリアルタイムでモニタリングできます。メトリクスは、リージョン、クラスター ID、HSM ID、およびクラスター ID ごとにグループ化できます。

Amazon CloudWatch メトリックスを使用すると、サービスに影響を及ぼす可能性のある潜在的な問題を警告するように Amazon CloudWatch アラームを設定できます。以下を監視するようにアラームを設定することをお勧めします。

- HSM のキー制限に近づく
- HSM の HSM セッション数の制限に近づいています
- HSM の HSM ユーザー数制限に近づいています。
- 同期の問題を特定するための HSM ユーザー数またはキー数の違い
- が問題を解決するまでクラスター AWS CloudHSM をスケールアップする異常な HSMs

詳細については、Amazon CloudWatch Logs と AWS CloudHSM 監査ログの使用を参照してください。

のクラスター AWS CloudHSM

クラスターは、 が同期 AWS CloudHSM を維持する個々のハードウェアセキュリティモジュール (HSM) のコレクションです。クラスター内にある HSM でタスクまたはオペレーションを行うと、そのクラスター内の他の HSM は、自動的に最新の状態に維持されます。

AWS CloudHSM クラスターは、 AWS CloudHSM コンソール、 AWS SDKsコマンドラインツールのいずれかから管理できます。詳細については、以下のトピックを参照してください。

クラスターを作成するには、「開始方法」を参照してください。

次のトピックでは、クラスターについて詳しく説明します。

トピック

- AWS CloudHSM クラスターアーキテクチャ
- AWS CloudHSM クラスターの同期
- AWS CloudHSM クラスターの高可用性とロードバランシング
- AWS CloudHSM クラスターモード
- の HSM タイプ AWS CloudHSM
- クライアント SDK を AWS CloudHSM クラスターに接続する
- AWS CloudHSM クラスターでの HSMsスケーリング
- AWS CloudHSM クラスターの削除
- バックアップからの AWS CloudHSM クラスターの作成
- クラスター HSM タイプの移行

AWS CloudHSM クラスターアーキテクチャ

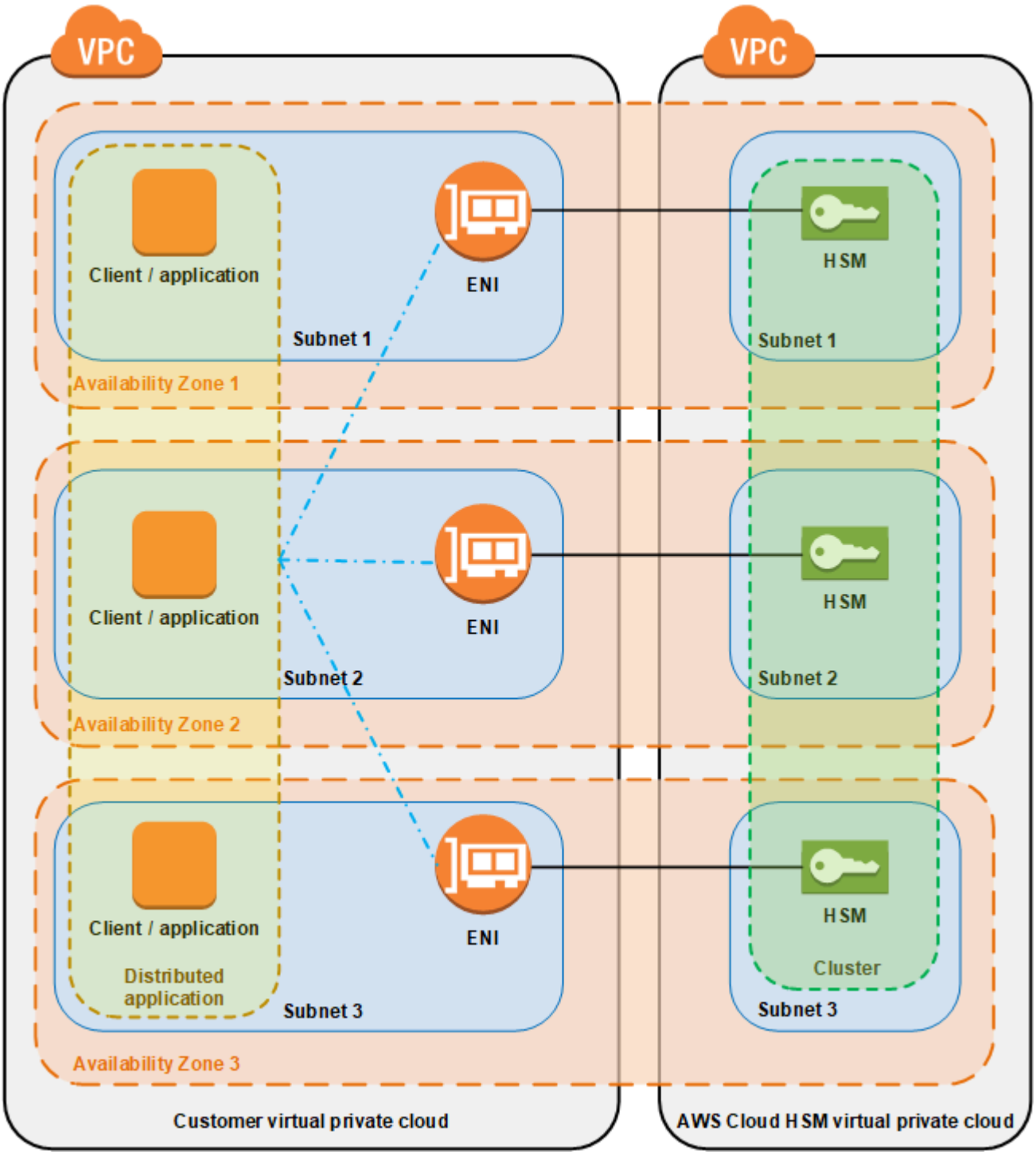
クラスターを作成するときは、 AWS アカウントに Amazon Virtual Private Cloud (VPC) を指定し、その VPC に 1 つ以上のサブネットを指定します。選択した AWS リージョンの各アベイラビリティーゾーン (AZ) に 1 つのサブネットを作成することをお勧めします。VPC を作成するときにプライベートサブネットを作成できます。詳細については の仮想プライベートクラウド (VPC) を作成する AWS CloudHSMを参照してください。

HSM を作成する度に、HSM のクラスターとアベイラビリティーゾーンを指定します。HSM を別々のアベイラビリティーゾーンに指定すると、いずれかのアベイラビリティーゾーンが使用できなくなった場合でも冗長性と高可用性を維持します。

HSM を作成すると、 は AWS アカウントの指定されたサブネットに Elastic Network Interface (ENI) AWS CloudHSM を配置します。Elastic Network Interface は、HSM とやり取りするためのインターフェイスです。HSM は、 が所有する AWS アカウントの別の VPC にあります AWS CloudHSM。HSM と対応するネットワークインターフェイスは、同じアベイラビリティーゾーンに存在します。

クラスター内の HSMs を操作するには、 AWS CloudHSM クライアントソフトウェアが必要です。通常、次の図に示すように、HSM ENI と同じ VPC にある Amazon EC2 インスタンス (クライアントインスタンス) でクライアントをインストールします。ただし、これは技術的には必要ありません。HSM ENI に接続できる限り、互換性のある任意のコンピュータでクライアントをインストールできます。クライアントは ENI を通じてクラスター内の個々の HSM と通信します。

次の図は、それぞれ VPC 内の異なるアベイラビリティーゾーンにある 3 つの HSMs を持つ AWS CloudHSM クラスターを示しています。



AWS CloudHSM クラスターの同期

AWS CloudHSM クラスターでは、 AWS CloudHSM は個々の HSMsのキーを同期させます。HSM 上でキーを同期するために必要な操作はありません。キーとは異なり、クラスター間で HSM ユーザーを同期するサーバー側のメカニズムはありません。 AWS CloudHSM CLI は HSMs 間でユー

ザーオペレーションのベストエフォート同期を実行しますが、オペレーションが部分的に失敗すると不整合が発生する可能性があります。ユーザーが同期しなくなると、`user list`コマンドに不整合が表示されます。詳細については、「クライアント SDK 5 ユーザーまたはポリシーに整合性のない値が含まれている」を参照してください。

クラスターに新しい HSM を追加すると、 は既存の HSM のすべてのキー、ユーザー、ポリシーのバックアップ AWS CloudHSM を作成します。次に、そのバックアップが新しい HSM に復元されます。これにより、2 つの HSM の同期が保たれます。

クラスター内の HSMs のキーが同期に失敗した場合、 によって AWS CloudHSM 自動的に再同期されます。これを有効にするために、 はアプライアンスユーザーの認証情報 AWS CloudHSM を使用します。このユーザーは、 が提供するすべての HSMs に存在し AWS CloudHSM 、アクセス許可が制限されています。HSM でオブジェクトのハッシュの取得と、マスク (暗号化) されたオブジェクトの抽出および挿入を行うことができます。 AWS は、ユーザーあるいはキーの表示や変更、およびこのキーを使用した一切の暗号化オペレーションを実行することはできません。

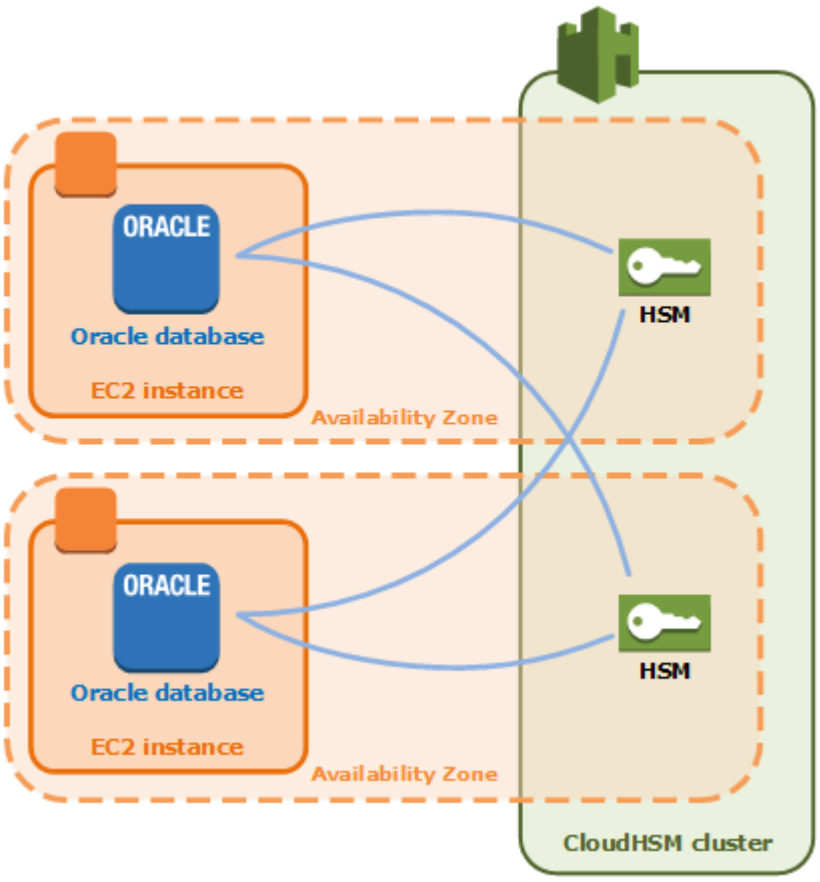
この自動再同期は、 キーにのみ適用されます。ユーザーとポリシー (mTLS 設定など) は自動的に再同期されません。CloudHSM CLI はHSMs 間でユーザーオペレーションとポリシーオペレーションのベストエフォート同期を実行しますが、不整合が発生する可能性があり、手動で解決する必要がある場合があります。詳細については、「クライアント SDK 5 ユーザーまたはポリシーに整合性のない値が含まれている」を参照してください。

AWS CloudHSM クラスターの高可用性とロードバランシング

複数の HSM を持つ AWS CloudHSM クラスターを作成すると、自動的に負荷分散が行われます。ロードバランシングは、追加の処理に対する HSM の容量に基づき、AWS CloudHSM クライアントによって、クラスター内のすべての HSM に暗号化オペレーションが分散されることを意味します。

異なる AWS アベイラビリティーゾーンに HSMs を作成すると、高可用性が自動的に取得されます。高可用性は、個々の HSM に単一障害点がないことにより、高い信頼性を取得できることを意味します。各クラスターには最低 2 つの HSMs があり、各 HSM は AWS リージョン内の異なるアベイラビリティーゾーンにあることをお勧めします。

たとえば、次の図では、Oracle データベースアプリケーションが 2 つの異なるアベイラビリティーゾーンに分散されています。データベースインスタンスは、各アベイラビリティーゾーンに HSM を含むクラスターにマスターキーを保存します。 は、キーを両方の HSMs AWS CloudHSM に自動的に同期して、すぐにアクセスでき冗長になるようにします。



AWS CloudHSM クラスターモード

AWS CloudHSM は、FIPS と非 FIPS の 2 つのモードでクラスターを提供します。FIPS モードでは、連邦情報処理標準 (FIPS) で検証されたキーとアルゴリズムのみを使用できます。非 FIPS モードは、FIPS の承認に関係なく AWS CloudHSM、 でサポートされているすべてのキーとアルゴリズムを提供します。

どのクラスタータイプと HSM タイプがニーズに合っているかを判断する前に、このページの詳細を確認してください。

> **Note**
>
> 2024 年 6 月 10 日より前に作成されたすべてのクラスターは FIPS モードで動作しており、HSM タイプは hsm1.medium です。

クラスターのモードと HSM タイプを確認するには、describe-clusters コマンドを使用します。

次の表に、各クラスター モードの主な違いを示します。

差別化機能	FIPS モード	非 FIPS モード
HSM タイプの互換性	hsm1.medium および hsm2m.medium で使用できます。	hsm2m.medium で使用できます。
バックアップ互換性	FIPS モードでクラスターをバックアップ復元するためにのみ使用できます。	非 FIPS モードで復元クラスターのバックアップにのみ使用できます。
キーの選択	FIPS 承認済みの メカニズムでキーの生成と使用をサポートします[1]。	他の非検証メカニズムに加えて、すべての FIPS 検証済みメカニズムでキーの生成と使用をサポートします。
アルゴリズム	FIPS 承認済みの AWS CloudHSM アルゴリズムをサポートします[1]。	FIPS 承認アルゴリズムと FIPS 承認 AWS CloudHSM アルゴリズムの両方をサポートします。

[1] 詳細については、「Deprecation notifications」を参照してください。

クラスター モードを選択する際、クラスターを作成すると、そのモード (FIPS または非 FIPS) を変更できないことに注意してください。要件に合った適切なモードを選択してください。

の HSM タイプ AWS CloudHSM

AWS CloudHSM には、hsm1.medium と hsm2m.medium の 2 つのハードウェアセキュリティモジュール (HSM) タイプも用意されています。どの HSM タイプがニーズに合っているかを判断する前に、このページの詳細を確認してください。

クラスターモードに加えて、 は hsm1.medium と hsm2m.medium の 2 つの HSM タイプ AWS CloudHSM を提供します。HSM の種類ごとに使用するハードウェアは異なり、各クラスターには 1 種類の HSM しか含めることができません。次の表はこの 2 つの主な相違点の一覧です。

差別化機能	hsm1.medium	hsm2m.medium
クラスターモードの互換性	FIPS モードでクラスターで使用できます。	FIPS モードまたは非 FIPS モードのクラスターで使用できます。
ネットワークタイプの互換性	利用不可	FIPS モードまたは非 FIPS モードのクラスターで使用できます。
バックアップ互換性	FIPS モードで hsm1.medium クラスターと hsm2m.medium クラスターへのバックアップと復元に使用できます。	hsm2m.medium クラスターのバックアップと復元にのみ使用できます。
キー容量	クラスターあたり 3,300 個。	全体で16,666個のキーをサポートします。非対称鍵は、クラスターあたり最大3,333個まで保存できます。
クライアント SDK	すべてのクライアント SDK をサポートします。	すべてのクライアント SDK をサポートします。
クライアント SDK のバージョン	SDK バージョン 3.1.0 以降と互換性があります。	Client SDK バージョン 5.9.0 以降と互換性があります。
利用可能なリージョン	CloudHSM は、どの AWS リージョンでも新しいクラスターの作成をサポートしなくなりました。詳細については、「非推奨通知」を参照してください。	CloudHSM が利用可能な AWS リージョン で利用できます。CloudHSM
パフォーマンス	各 HSM タイプのパフォーマンスを確認するには、AWS CloudHSM パフォーマンス情報 を参照してください。	

差別化機能	hsm1.medium	hsm2m.medium
証明書	FIPS 140-2、PCI DSS、PCI PIN、SOC2、PCI-3DS に準拠しています。	FIPS 140-3、PCI DSS、PCI PIN、SOC2、PCI-3DS に準拠しています。

クライアント SDK を AWS CloudHSM クラスターに接続する

クライアント SDK 5 またはクライアント SDK 3 のいずれかを用いてクラスターに接続するには、まず 2 つの操作を行う必要があります。

- EC2 インスタンス上に発行証明書を配置する
- クライアント SDK をクラスターにブートストラップする

各 EC2 インスタンス上に発行証明書を配置する

クラスターの初期化時には、発行証明書を作成します。クラスターに接続する各 EC2 インスタンス上のプラットフォームのために、デフォルトの場所への発行証明書をコピーします。

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

発行証明書の場所を指定する

クライアント SDK 5 を用いて、構成ツールを使用して発行証明書の場所を指定できます。

PKCS #11 library

Linux クライアント SDK 5 の発行証明書を配置する

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate
 file>
```

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --hsm-ca-
cert <customerCA certificate file>
```

OpenSSL Dynamic Engine

Linux クライアント SDK 5 の発行証明書を配置する

- 構成ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate
 file>
```

OpenSSL Dynamic Engine Provider

Linux クライアント SDK 5 の発行証明書を配置する

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --hsm-ca-cert <customerCA
 certificate file>
```

Key Storage Provider (KSP)

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --hsm-ca-
cert <customerCA certificate file>
```

JCE provider

Linux クライアント SDK 5 の発行証明書を配置する

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate
 file>
```

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --hsm-ca-
cert <customerCA certificate file>
```

CloudHSM CLI

Linux クライアント SDK 5 の発行証明書を配置する

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate
 file>
```

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --hsm-ca-
cert <customerCA certificate file>
```

さらなる詳細については、Configure Tool を参照してください。

クラスターの初期化、または証明書の作成と署名の詳細については、Initilize the Cluster を参照してください。

クライアント SDK をブートストラップする

ブートストラッププロセスは、使用しているクライアント SDK のバージョンによって異なりますが、クラスター内のいずれかのハードウェアセキュリティモジュール (HSM) の IP アドレスが必要です。クラスターに添付されている任意の HSM の IP アドレスを使用できます。クライアント SDK が接続すると、あらゆる追加の HSM の IP アドレスを取得し、ロードバランシングとクライアント側のキー同期操作を実行します。

クラスターのために IP アドレスを取得するには

HSM の IP アドレスを取得するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. AWS リージョンを変更するには、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. クラスターの詳細ページを開くには、クラスターテーブルでクラスター ID を選択します。
4. IP アドレスを取得するには、HSM タブに移動します。IPv4 クラスターの場合は、ENI IPv4 アドレスにリストされているアドレスを選択します。デュアルスタッククラスターの場合は、ENI IPv4 アドレスまたは ENI IPv6 アドレスのいずれかを使用します。

HSM の IP アドレスを取得する (AWS CLI)

- describe-clusters から AWS CLIコマンドを実行して、HSM の IP アドレスを取得します。コマンドの出力では、HSM の IP アドレスは `EniIp` および `EniIpV6` の値です (デュアルスタッククラスターの場合)。

```
$ aws cloudhsmv2 describe-clusters
{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
                    "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733",
```

```
...
```

ブートストラップのさらなる詳細については、構成ツール を参照してください。

クライアント SDK 5 のブートストラップ

PKCS #11 library

クライアント SDK 5 用のための Linux の EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP
 addresses>
```

OpenSSL Dynamic Engine

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

OpenSSL Dynamic Engine Provider

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider -a <HSM IP addresses>
```

Key Storage Provider (KSP)

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" -a <HSM IP
 addresses>
```

JCE provider

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP
 addresses>
```

CloudHSM CLI

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the
 HSMs>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI
 IPv4 / IPv6 addresses of the HSMs>
```

> **Note**
>
> `--cluster-id` パラメータは `-a <HSM_IP_ADDRESSES>` の代わりに使用できます。`--cluster-id` の使用要件については、「AWS CloudHSM クライアント SDK 5 設定ツール」を参照してください。

クライアント SDK 3 をブートストラップするには

クライアント SDK 3 用の Linux の EC2 インスタンスをブートストラップするには

- configure でクラスター内の HSM の IP アドレスを指定します。

```
sudo /opt/cloudhsm/bin/configure -a <IP address>
```

クライアント SDK 3 用の Windows EC2 インスタンスをブートストラップするには

- configure でクラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM IP address>
```

さらなる設定の詳細については、??? を参照してください。

AWS CloudHSM クラスターでの HSMsスケーリング

AWS CloudHSM クラスターをスケールアップまたはスケールダウンするには、AWS CloudHSM コンソール、いずれかの SDK、またはコマンドラインツールを使用して HSMs を追加または削除します。 AWS SDKs クラスターの負荷テストを行って予測すべきピーク負荷を決定し、高可用性を確保するためにクラスターに HSM を 1 つ追加することを推奨します。

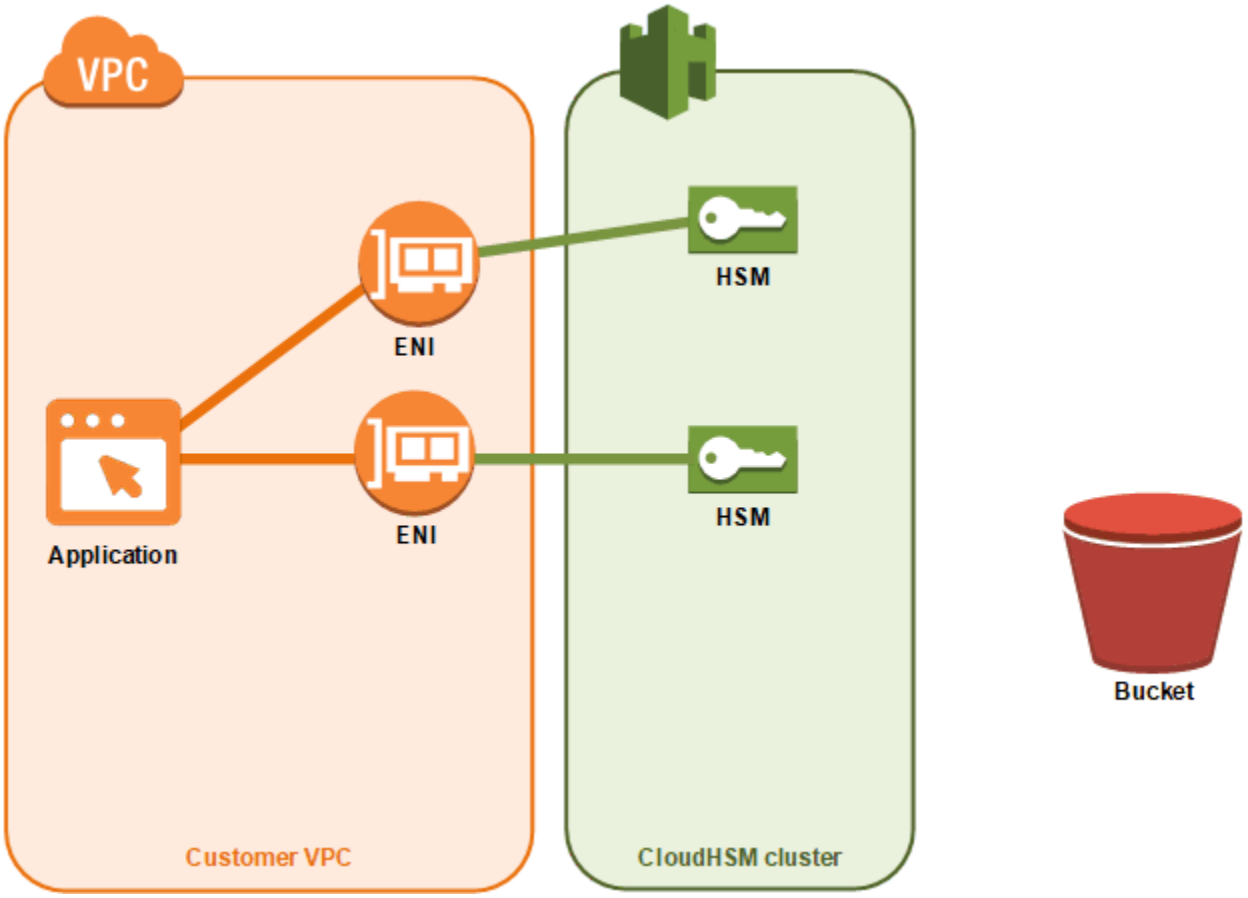
トピック

- AWS CloudHSM クラスターへの HSM の追加

- AWS CloudHSM クラスターからの HSM の削除

AWS CloudHSM クラスターへの HSM の追加

次の図は、クラスターに HSM を追加したときに発生するイベントを示しています。



1. 新しい HSM をクラスターに追加します。以下の手順では、この操作を AWS CloudHSM コンソール、AWS Command Line Interface (AWS CLI)、および AWS CloudHSM API で行う方法を示します。

   これは、ユーザーが行う唯一のアクションです。残りのイベントは自動的に実行されます。

2. AWS CloudHSM は、クラスター内の既存の HSM のバックアップコピーを作成します。詳細については、「バックアップ」を参照してください。

3. AWS CloudHSM は、バックアップを新しい HSM に復元します。これにより、HSM はクラスター内の他のインスタンスと同期されます。

4. クラスター内の既存の HSMs は、クラスターに新しい HSM があることを AWS CloudHSM クライアントに通知します。

5. クライアントは、新しい HSM クライアントへの接続を確立します。

HSM を追加するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. HSM を追加する先のクラスターを選択します。
3. [HSMs] タブで [Create HSM] を選択します。
4. 作成中の HSM のアベイラビリティーゾーン (AZ) を選択します。次に [作成] を選択します。

HSM を追加するには (AWS CLI)

- コマンドプロンプトで、create-hsm コマンドを発行し、作成する HSM 用のクラスター ID とアベイラビリティーゾーンを指定します。該当するクラスターのクラスター ID がわからない場合は、describe-clusters コマンドを発行します。アベイラビリティーゾーンは、us-east-2a、us-east-2b などの形式で指定します。

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-
zone <Availability Zone>
{
    "Hsm": {
        "State": "CREATE_IN_PROGRESS",
        "ClusterId": "cluster-5a73d5qzrdh",
        "HsmId": "hsm-lgavqitns2a",
        "SubnetId": "subnet-0e358c43",
        "AvailabilityZone": "us-east-2c",
        "EniId": "eni-bab18892",
        "EniIp": "10.0.3.10",
        "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733"
    }
}
```

HSM (AWS CloudHSM API) を追加するには

- CreateHsm リクエストを送信し、作成する HSM 用のクラスター ID とアベイラビリティーゾーンを指定します。

AWS CloudHSM クラスターからの HSM の削除

AWS CloudHSM コンソール、、AWS CLIまたは AWS CloudHSM API を使用して HSM を削除できます。

HSM を削除するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. 削除する HSM が含まれているクラスターを選択します。
3. [HSMs] タブで、削除する HSM を選択します。次に、[Delete HSM] を選択します。
4. HSM を削除することを確定します。その後、[削除] をクリックします。

HSM を削除するには (AWS CLI)

- コマンドラインプロンプトで、delete-hsm コマンドを発行します。削除する HSM が含まれているクラスターの ID と、以下のいずれかの HSM 識別子を渡します。
  - HSM ID (`--hsm-id`)
  - HSM IP アドレス (`--eni-ip`)
  - HSM の Elastic Network Interface ID (`--eni-id`)

  これらの識別子の値がわからない場合は、describe-clusters コマンドを発行します。

```
$ aws cloudhsmv2 delete-hsm --cluster-id <cluster ID> --eni-ip <HSM IP address>
{
    "HsmId": "hsm-lgavqitns2a"
}
```

HSM (AWS CloudHSM API) を削除するには

- DeleteHsm リクエストを送信し、クラスター ID と削除する HSM の識別子を指定します。

AWS CloudHSM クラスターの削除

クラスターを削除する前に、すべての HSM をクラスターから削除する必要があります。詳細については、「AWS CloudHSM クラスターからの HSM の削除」を参照してください。

すべての HSMs を削除したら、 AWS CloudHSM コンソール、 AWS Command Line Interface (AWS CLI)、または AWS CloudHSM API を使用してクラスターを削除できます。

クラスターを削除するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. 削除するクラスターを選択します。次に、[クラスターの削除] を選択します。
3. クラスターを削除することを確認し、[削除] を選択します。

クラスターを削除するには (AWS CLI)

- コマンドプロンプトで、delete-cluster コマンドを発行し、削除するクラスターの ID を渡します。クラスター ID がわからない場合は、describe-clusters コマンドを発行します。

```
$ aws cloudhsmv2 delete-cluster --cluster-id <cluster ID>
{
    "Cluster": {
        "Certificates": {
            "ClusterCertificate": "<certificate string>"
        },
        "SourceBackupId": "backup-rtq2dwi2gq6",
        "SecurityGroup": "sg-40399d28",
        "CreateTimestamp": 1504903546.035,
        "SubnetMapping": {
            "us-east-2a": "subnet-f1d6e798",
            "us-east-2c": "subnet-0e358c43",
            "us-east-2b": "subnet-40ed9d3b"
        },
        "ClusterId": "cluster-kdmrayrc7gi",
        "VpcId": "vpc-641d3c0d",
        "State": "DELETE_IN_PROGRESS",
        "HsmType": "hsm1.medium",
        "StateMessage": "The cluster is being deleted.",
        "Hsms": [],
        "BackupPolicy": "DEFAULT"
    }
}
```

AWS CloudHSM クラスターを削除するには (API)

- DeleteCluster リクエストを送信し、削除するクラスターの ID を指定します。

バックアップからの AWS CloudHSM クラスターの作成

バックアップから AWS CloudHSM クラスターを復元するには、このトピックの手順に従います。クラスターには、バックアップにあったものと同じユーザー、キーマテリアル、証明書、設定、およびポリシーが含まれます。バックアップの管理に関する詳細については、クラスターのバックアップを参照してください。

バックアップからのクラスターの作成 (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. [クラスターを作成] を選択します。
3. [Cluster configuration] セクションで、以下の操作を実行します。
   a. [VPC] で、作成するクラスターの VPC を選択します。
   b. [AZ(s)] で、クラスターに追加する各アベイラビリティーゾーンのプライベートサブネットを選択します。
   c. [ネットワークタイプ] で、HSM が接続に使用する IP プロトコルを選択します。
4. [Cluster source] セクションで、以下の操作を行います。
   a. [Restore cluster from existing backup] を選択します。
   b. 復元するバックアップを選択します。
5. [次へ: レビュー] を選択します。
6. クラスター設定を確認し、[Create cluster] を選択します。
7. サービスがバックアップを保持する期間を指定します。

   デフォルトの保存期間である 90 日を受け入れるか、7 ~ 379 日の間に新しい値を入力します。このサービスは、ここで指定した値よりも古いこのクラスター内のバックアップを自動的に削除します。これは後で変更できます。詳細については、「バックアップ保持の設定」を参照してください。
8. [Next] (次へ) を選択します。
9. (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。クラスターに複数のタグを追加するには、タグの追加 を選択します。

10. [Review] (レビュー) を選択します。
11. クラスター設定を確認し、[Create cluster (クラスターの作成)] を選択します。

> **Tip**
>
> 復元したバックアップと同じユーザー、キーマテリアル、証明書、設定、およびポリシーを含む HSM をこのクラスターに作成するには、クラスターに HSM を追加します。

バックアップからのクラスターの作成 (AWS CLI)

バックアップ ID を判別するには、describe-backups コマンドを発行します。

- コマンドラインプロンプトで、create-cluster コマンドを発行します。HSM インスタンスタイプ、HSM を作成するサブネットのサブネット ID、および復元するバックアップのバックアップ ID を指定します。

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
                                --subnet-ids <subnet ID 1> <subnet ID 2> <subnet ID
 N> \
                                --source-backup-id <backup ID>
                                --mode <FIPS> \
                                --network-type <IPV4>
{
    "Cluster": {
        "HsmType": "hsm2m.medium",
        "VpcId": "vpc-641d3c0d",
        "Hsms": [],
        "State": "CREATE_IN_PROGRESS",
        "SourceBackupId": "backup-rtq2dwi2gq6",
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
        "NetworkType": "IPV4",
        "SecurityGroup": "sg-640fab0c",
        "CreateTimestamp": 1504907311.112,
        "SubnetMapping": {
            "us-east-2c": "subnet-0e358c43",
            "us-east-2a": "subnet-f1d6e798",
```

```
            "us-east-2b": "subnet-40ed9d3b"
        },
        "Certificates": {
            "ClusterCertificate": "<certificate string>"
        },
        "ClusterId": "cluster-jxhlf7644ne"
    }
}
```

バックアップからクラスターを作成する (AWS CloudHSM API)

API を使用してバックアップからクラスターを作成する方法については、次のトピックを参照してください。

- CreateCluster

クラスター HSM タイプの移行

AWS CloudHSM では、既存のクラスターの HSM タイプを変更することができます。このページの表を確認して、HSM タイプの変更が許可されているかどうかを確認します。

サポートされている HSM のタイプおよび機能については「の HSM タイプ AWS CloudHSM」を参照してください。

> **Note**
>
> このオペレーション中にクラスターの FIPS モードを変更することはできません。

から	まで	Comment
hsm1.medium	hsm2m.medium	許可されています
hsm2m.medium	hsm1.medium	条件付き。移行の開始から 24 時間以内に hsm2m.medium から hsm1.medium にロールバックできます。

トピック

- hsm1.medium から hsm2m.medium への移行

hsm1.medium から hsm2m.medium への移行

AWS CloudHSM クラスターを hsm1.medium から hsm2m.medium に移行できます。このトピックでは、前提条件、移行プロセス、ロールバック手順について説明します。

移行を開始する前に、アプリケーションが 高可用性対応のクラスターをアーキテクチャー の推奨事項に従っていることを確認してください。これにより、プロセス中のダウンタイムを回避できます。

> **移行の自動更新**
>
> 2026 年 1 月 20 日をもって、hsm2m.medium への自動移行が開始されました。

hsm1.medium から hsm2m.medium への移行プロセスの概要

コンソール AWS CloudHSM 、、または AWS CloudHSM API を使用して AWS CLI移行を開始できます。どこから開始しても、 AWS CloudHSM クラスター移行は `modify-cluster` API エンドポイントを使用します。または、AWS CloudHSM はユーザーに代わってクラスターを自動的に移行します。移行が開始されると、クラスター全体が書き込み制限モードになります。詳細については、「クラスターの書き込み制限モード」を参照してください。

影響を最小限に抑えるために、 は HSMs hsm1.medium から hsm2m.medium に一度に 1 つずつ AWS CloudHSM 変更します。置き換え後の HSM は同じ IP アドレスを維持するため、移行中も移行後も設定変更は不要です。

移行は次の手順で実行されます。

1. 最初の HSM を移行する前に、 はクラスター全体のフルバックアップ AWS CloudHSM を作成します。
2. このバックアップを使用して、 はリクエストされたタイプ (hsm2m.medium) の新しい HSM AWS CloudHSM を作成し、最初の HSM を置き換えます。
3. 後続の各 HSM を移行する前に、 はクラスター全体の新しいフルバックアップ AWS CloudHSM を作成します。
4. AWS CloudHSM は、クラスター内の HSM ごとにステップ 2 と 3 を繰り返し、一度に 1 つの HSM を移行します。

5. 各 HSM の移行には、約 30 分かかります。

AWS CloudHSM はクラスターの状態を監視し、移行プロセス全体で検証を実行します。がエラーの増加 AWS CloudHSM を検出した場合、または検証チェックが失敗した場合、自動的に移行を停止し、クラスターを元の HSM タイプに戻します。移行開始後 24 時間以内であれば、手動でロールバックすることも可能です。ロールバックを実行する前に、「HSM タイプのロールバックに関する考慮事項」を参照してください。

hsm2m.medium に移行するための前提条件

hsm2m.medium に移行するには、既存の AWS CloudHSM クラスターがこれらの要件を満たしている必要があります。検証チェックのいずれかの条件を満たさない場合、 AWS CloudHSM は自動的にクラスターを元の HSM タイプに戻します。

移行に関するよくある問題については、「???」を参照してください。

- 過去 7 日間:
  - すべてのクライアント接続で SDK 5.9 以降が使用されている。
    - ECDSA Verify を実行する場合、すべてのクライアント接続が SDK 5.13 以上を使用している。
  - AWS CloudHSM インスタンスは、サポートされている機能のみを使用しています (非推奨の機能はありません) 。詳細については、「非推奨通知」を参照してください。
  - 過去 7 日以内に、クラスター内の少なくとも 1 台の HSM に対して、SDK を使用した接続が行われている。
- クラスターが ACTIVE 状態である。
- クラスターに含まれる HSM が 27 台以下である。
- 移行中に HSM オペレーションのエラーレートが増加しない。

> **Note**
>
> トークンキーのワークロードを持つお客様が移行できなかった以前の制限は解除されました。

> **SDK 接続要件の免除**
>
> クラスターに過去 14 日間 SDK またはクライアント接続がない場合、クラスターは 7 日間の SDK 接続要件から除外されます。

クラスターの書き込み制限モード

クラスターが移行を開始すると、制限付き書き込みモードになります。HSM の状態を変更する可能性のあるオペレーションは拒否されます。読み取りオペレーションはすべて影響を受けません。

移行中、アプリケーションが次のオペレーションを実行しようとすると、HSM からエラーが返されます。

- トークンキーの生成および削除 (セッションキーのワークロードは引き続き動作します)。
- ユーザーの作成、削除、変更のすべて。
- クォーラムオペレーション。
- キー属性の変更など、HSM 内のキーの変更。
- mTLS の登録。

AWS CloudHSM また、 は移行中にクラスターを `MODIFY_IN_PROGRESS`状態にします。この期間中は、クラスターに HSM を追加したり、クラスターから HSM を削除したりすることはできません。

移行の開始

クラスター移行プロセスでは、クラスター内の HSM を 1 台ずつ順番に置き換えていきます。所要時間は、クラスター内の HSM の台数に依存します。平均して HSM 1 台あたり約 30 分 かかります。クラスター内の各 HSM の HSM タイプをモニタリングすることで、何台の HSM が新しいタイプへ移行済みかを確認し、進行状況を追跡できます。

Console

HSM タイプを変更するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。

2. 変更するクラスターの ID の横のラジオボタンを選択します。
3. [アクション] メニューから、Modify HSM Type を選択し、目的の HSM タイプを選択します。

この手順を実行すると、クラスターは MODIFY_IN_PROGRESS 状態になります。移行が完了すると、クラスターは ACTIVE 状態に戻ります。

AWS CLI

HSM タイプを変更するには (AWS CLI)

- コマンドラインプロンプトで、 modify-cluster コマンドを実行します。クラスター ID と目的の HSM タイプを指定します。

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> --hsm-type <HSM Type>


{
    "Cluster": {
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
        "VpcId": "vpc-50ae0636",
        "SubnetMapping": {
            "us-west-2b": "subnet-49a1bc00",
            "us-west-2c": "subnet-6f950334",
            "us-west-2a": "subnet-fd54af9b"
        },
        "SecurityGroup": "sg-6cb2c216",
        "HsmType": "hsm2m.medium",
        "HsmTypeRollbackExpiration": 1730383180.000,
        "Certificates": {},
        "State": "MODIFY_IN_PROGRESS",
        "Hsms": [],
        "ClusterId": "cluster-igklspoyj5v",
        "ClusterMode": "FIPS",
        "CreateTimestamp": 1502423370.069
    }
}
```

この手順を実行すると、クラスターは MODIFY_IN_PROGRESS 状態になります。移行が完了すると、クラスターは ACTIVE 状態に戻ります。

AWS CloudHSM API

HSM タイプを変更するには (AWS CloudHSM API)

- ModifyCluster リクエストを送信します。クラスターの ID と目的の HSM タイプを指定します。

この手順を実行すると、クラスターは MODIFY_IN_PROGRESS 状態になります。移行が完了すると、クラスターは ACTIVE 状態に戻ります。

移行のロールバック

AWS CloudHSM は、エラー率の上昇をモニタリングし、移行全体を通して継続的な検証チェックを実行します。がサービス品質の低下または検証の失敗 AWS CloudHSM を検出すると、クラスターの元の HSM タイプへのロールバックが自動的に開始されます。ロールバック中は、クラスター内の各 HSM に対して次の処理が行われます。

- AWS CloudHSM は、その HSM の移行の開始時に取得したバックアップを使用します。
- すべての HSM が元のタイプに戻るまで、HSM を 1 台ずつ順番に置き換えます。
- ロールバック処理の全期間を通じて、クラスターは書き込み制限モードのまま維持されます。

移行開始から 24 時間以内であれば、移行をロールバックできます。ロールバックの期限を確認するには:

1. describe-clusters コマンドを実行します。
2. HsmTypeRollbackExpiration の値を探します。このタイムスタンプはロールバックの期限です。

ロールバックする場合は、この期限までにロールバックしてください。ロールバックでは、元の HSM タイプの最新のバックアップが使用されます。

> **Warning**
>
> 移行完了後のロールバックには注意が必要です。移行を完了し、 AWS CloudHSM を使用して新しいキーまたはユーザーを作成すると、ロールバックによってデータが失われる可能性があります。たとえば、hsm2m.medium で作成されたキーは、ロールバックが完了したときに hsm1.medium で使用できない場合があります。ロールバック後のデータ損失を軽減する方法については、ロールバック後のデータ同期を参照してください。

Console

HSM タイプをロールバックするには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. ロールバックしたいクラスターの ID を選択します。
3. [アクション] メニューから、`Modify HSM Type` を選択し、元の HSM タイプを選択します。

この手順を実行すると、クラスターは `ROLLBACK_IN_PROGRESS` 状態になります。ロールバックが完了すると、クラスターは `ACTIVE` 状態に戻ります。

AWS CLI

HSM タイプをロールバックするには (AWS CLI)

- コマンドラインプロンプトで、 modify-cluster コマンドを実行します。クラスター ID と元の HSM タイプを指定します。

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> --hsm-type <HSM Type>

{
 "Cluster": {
     "BackupPolicy": "DEFAULT",
     "BackupRetentionPolicy": {
         "Type": "DAYS",
         "Value": 90
      },
     "VpcId": "vpc-50ae0636",
     "SubnetMapping": {
```

```
            "us-west-2b": "subnet-49a1bc00",
            "us-west-2c": "subnet-6f950334",
            "us-west-2a": "subnet-fd54af9b"
        },
        "SecurityGroup": "sg-6cb2c216",
        "HsmType": "hsm1.medium",
        "HsmTypeRollbackExpiration": 1730383180.000,
        "Certificates": {},
        "State": "ROLLBACK_IN_PROGRESS",
        "Hsms": [],
        "ClusterId": "cluster-igklspoyj5v",
        "ClusterMode": "FIPS",
        "CreateTimestamp": 1502423370.069
    }
}
```

この手順を実行すると、クラスターは ROLLBACK_IN_PROGRESS 状態になります。ロールバックが完了すると、クラスターは ACTIVE 状態に戻ります。

AWS CloudHSM API

HSM タイプをロールバックするには (AWS CloudHSM API)

- ModifyCluster リクエストを送信します。クラスターの ID と元の HSM タイプを指定します。

この手順を実行すると、クラスターは ROLLBACK_IN_PROGRESS 状態になります。ロールバックが完了すると、クラスターは ACTIVE 状態に戻ります。

ロールバック後のデータ同期

移行中、HSM は書き込み制限モードであるため、HSM の状態を変更することはできません。この期間中 (クラスターが MODIFY_IN_PROGRESS の間) にロールバックを行うと、クラスターの内容は元のクラスターと完全に同一の状態になります。

クラスターが ACTIVE 状態に戻ると、書き込み制限モードは解除されます。ACTIVE 状態の間にキーやユーザーを作成し、その後ロールバックを実行した場合、そのキーやユーザーはロールバック後のクラスターには存在しません。

ユーザーの同期に対応するには、CloudHSM CLI を使用したユーザー管理で、ロールバック後のクラスター上に不足しているユーザーを再作成します。これは、`user replicate` コマンドが hsm2m.medium から hsm1.medium へのユーザー同期をサポートしていないため、ユーザーを手動で再作成する必要があるためです。詳細については、ユーザーレプリケートに関する既知の問題を参照してください。

キーの同期に対応するには、key replicate コマンドを使用して、2 つのクラスター間でキーをレプリケートします。CloudHSM CLI をインストールしていない場合は、??? の手順を参照してください。

ロールバック後にキーを同期するには

ロールバック完了後、次の手順に従います。ここでは以下の用語を使用します。

- cluster-1: ロールバック後のクラスター (現在 hsm1.medium)
- cluster-2: 新たに作成する一時的な hsm2m.medium クラスター

1. cluster-1 の最新の hsm2m.medium バックアップを使用して、新しい hsm2m.medium クラスター (cluster-2) を作成します。

```
aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
                              --subnet-ids <subnet ID 1> <subnet ID 2> <subnet ID
 N> \
                              --source-backup-id <backup ID>
                              --mode <FIPS>
```

2. cluster-2 に HSM を作成します。

```
aws cloudhsmv2 create-hsm --cluster-id <cluster-2 ID>
```

3. cluster-2 で、レプリケーションが必要なキーを一覧表示します。

```
cloudhsm-cli key list --cluster-id <cluster-2 ID>
```

4. cluster-2 から cluster-1 へ、各キーをレプリケートします。

```
cloudhsm-cli key replicate --source-cluster-id <cluster-2 ID> \
                        --destination-cluster-id <cluster-1 ID> \
                        --filter attr.label=<key ID>
```

5. コピーが必要なすべてのキーについて、手順 4 を繰り返します。
6. cluster-2 の HSM を削除します。

```
aws cloudhsmv2 delete-hsm --cluster-id <cluster-2 ID> --hsm-id <HSM ID>
```

7. cluster-2 を削除します。

```
aws cloudhsmv2 delete-cluster --cluster-id <cluster-2 ID>
```

の HSM ユーザー AWS CloudHSM

AWS CloudHSM クラスターを暗号化処理に使用する前に、クラスターのハードウェアセキュリティモジュール (HSM) でユーザーとキーを作成する必要があります。

> **Note**
>
> HSM ユーザーは IAM ユーザーとは異なります。正しい認証情報を持つ IAM ユーザーは、AWS API を介してリソースを操作することで HSM を作成できます。HSM を作成したら、HSM ユーザー認証情報を使用して HSM でのオペレーションを認証する必要があります。

では AWS CloudHSM、CloudHSM CLI または CloudHSM Management Utility (CMU) コマンドラインツールを使用して、HSM のユーザーを作成および管理する必要があります。CloudHSM CLI は最新の SDK バージョンシリーズ で使用するように設計されていますが、CMU は 以前の SDK バージョンシリーズ で使用するように設計されています。

AWS CloudHSMでの HSM ユーザーの管理の詳細については、次のトピックを参照してください。クォーラム認証 (M of N アクセスコントロールとも呼ばれます) の使用方法も学習できます。

トピック

- CloudHSM CLI による HSM ユーザー管理
- CloudHSM 管理ユーティリティ (CMU) による HSM ユーザー管理

CloudHSM CLI による HSM ユーザー管理

でハードウェアセキュリティモジュール (HSM) ユーザーを管理するには AWS CloudHSM、 管理者のユーザー名とパスワードを使用して HSM にログインする必要があります。管理者のみユーザーを管理できます。HSM には、admin という名前のデフォルト管理者が含まれています。admin クラスターのアクティブ化 の際に必要なパスワードを設定しました。

このトピックでは、CloudHSM CLI で HSM ユーザーの管理についてステップバイステップの手順と詳細を説明します。

トピック

- CloudHSM CLI でのユーザー管理の前提条件

- CloudHSM CLI の HSM ユーザータイプ
- CloudHSM CLI の HSM ユーザーアクセス許可テーブル
- CloudHSM CLI で HSM ユーザー管理者を作成する
- CloudHSM CLI で HSM Crypto User を作成する
- CloudHSM CLI でクラスター内のすべての HSM ユーザーを一覧表示する
- CloudHSM CLI で HSM ユーザーパスワードを変更する
- CloudHSM CLI で HSM ユーザーを削除する
- CloudHSM CLI で HSM ユーザーの MFA を管理する
- CloudHSM CLI を使用したクォーラム認証の管理 (M of N アクセスコントロール)

CloudHSM CLI でのユーザー管理の前提条件

CloudHSM CLI を使用して でハードウェアセキュリティモジュール (HSM) ユーザーを管理する前に AWS CloudHSM、以下の前提条件を満たす必要があります。以下のトピックでは、CloudHSM CLI の基本的な使い方について説明します。

トピック

- で HSM の IP アドレスを取得する AWS CloudHSM
- CloudHSM CLI をダウンロード

で HSM の IP アドレスを取得する AWS CloudHSM

CloudHSM CLI を使用する場合、設定ツールでローカル設定を更新する必要があります。CloudHSM CLI で設定ツールを実行する手順については、「コマンドラインインターフェイス (CLI) AWS CloudHSM の開始方法」を参照してください。-a パラメータには、クラスター内の HSM の IP アドレスを追加する必要があります。複数の HSM をお持ちの方は、任意の IP アドレスを使用できます。これで確実に CloudHSM CLI がクラスター全体に加えた変更を伝播できます。CloudHSM CLI はローカルファイルを使用してクラスター情報を追跡することに注意してください。特定のホストから CloudHSM CLI を最後に使用後にクラスターが変更されている場合、該当するホストに保存されているローカル設定ファイルにそれらの変更を追加する必要があります。CloudHSM CLI 使用中は HSM を決して削除しないでください。

HSM の IP アドレスを取得するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。

2. AWS リージョンを変更するには、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. クラスターの詳細ページを開くには、クラスターテーブルでクラスター ID を選択します。
4. IP アドレスを取得するには、HSM タブに移動します。IPv4 クラスターの場合は、ENI IPv4 アドレスにリストされているアドレスを選択します。デュアルスタッククラスターの場合は、ENI IPv4 アドレスまたは ENI IPv6 アドレスのいずれかを使用します。

HSM の IP アドレスを取得する (AWS CLI)

- describe-clusters から AWS CLIコマンドを実行して、HSM の IP アドレスを取得します。コマンドの出力では、HSM の IP アドレスは `EniIp` および `EniIpV6` の値です (デュアルスタッククラスターの場合)。

```
$ aws cloudhsmv2 describe-clusters
{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
                    "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733",
...
```

CloudHSM CLI をダウンロード

CloudHSM CLI の最新バージョンは、クライアント SDK 5 の HSM ユーザー管理タスクに使用できます。CloudHSM CLI をダウンロードしてインストールするには、「CloudHSM CLI のインストールと設定」の指示に従ってください。

CloudHSM CLI の HSM ユーザータイプ

ハードウェアセキュリティモジュール (HSM) で実行するほとんどのオペレーションには、 AWS CloudHSM HSM ユーザーの認証情報が必要です。HSM は各 HSM ユーザーを認証し、各 HSM ユー

ザーに設定されている タイプ により、ユーザーとして HSM で実行できるオペレーションが決定されます。

> **Note**
>
> HSM ユーザーは IAM ユーザーとは異なります。正しい認証情報を持つ IAM ユーザーは、AWS API を介してリソースを操作することで HSM を作成できます。HSM を作成したら、HSM ユーザー認証情報を使用して HSM でのオペレーションを認証する必要があります。

ユーザータイプ

- 非アクティブ管理者
- 管理者
- Crypto User (CU)
- Appliance User (AU)

非アクティブ管理者

CloudHSM CLI では、非アクティブ化された管理者は、 AWS CloudHSM クラスター内のアクティブ化されたことのない最初の HSM にのみ存在する一時的なユーザーです。クラスターをアクティブ化する には、CloudHSM CLI で cluster activate コマンドを実行します。このコマンドを実行すると、非アクティブ化された管理者にはパスワードの変更を求めるプロンプトが表示されます。パスワードを変更すると、非アクティブ化された管理者は管理者になります。

管理者

CloudHSM CLI では、管理者はユーザー管理オペレーションを実行できます。たとえば、ユーザーの作成および削除と、ユーザーパスワードの変更を行うことなどができます。管理者の詳細については、「CloudHSM CLI の HSM ユーザーアクセス許可テーブル」を参照してください。

Crypto User (CU)

Crypto User (CU) は、以下のキー管理および暗号化のオペレーションを行うことができます。

- キー管理 - 暗号化キーの作成、削除、共有、インポート、エクスポートを行います。
- 暗号化オペレーション - 暗号化キーを使用して、暗号化、復号、署名、検証などを行います。

詳細については、 を参照してくださいCloudHSM CLI の HSM ユーザーアクセス許可テーブル。

Appliance User (AU)

アプライアンスユーザー (AU) は、クラスターの HSMs でクローン作成および同期オペレーションを実行できます。 は AU AWS CloudHSM を使用して、 AWS CloudHSM クラスター内の HSMsを同期します。AU は が提供するすべての HSMs に存在し AWS CloudHSM、アクセス許可が制限されています。詳細については、「CloudHSM CLI の HSM ユーザーアクセス許可テーブル」を参照してください。

AWS は HSMs でオペレーションを実行できません。 はユーザーまたはキーを表示または変更 AWS できず、これらのキーを使用して暗号化オペレーションを実行できません。

CloudHSM CLI の HSM ユーザーアクセス許可テーブル

以下の表は、 AWS CloudHSMでオペレーションを実行できる HSM ユーザーまたはセッションのタイプ別にソートされたハードウェアセキュリティモジュール (HSM) オペレーションを示しています。

	管理者	Crypto User (CU)	Appliance User (AU)	未認証セッション
基本的なクラスター情報を取得する[1]	✓ はい	✓ はい	✓ はい	✓ はい
自分のパスワードを変更する	✓ はい	✓ はい	✓ はい	該当しない
ユーザーのパスワードを変更する	✓ はい	✗ いいえ	✗ いいえ	✗ いいえ
ユーザーを追加、削除する	✓ はい	✗ いいえ	✗ いいえ	✗ いいえ
同期のステータスを取得する[2]	✓ はい	✓ はい	✓ はい	✗ いいえ

	管理者	Crypto User (CU)	Appliance User (AU)	未認証セッション
マスクされたオブジェクトを抽出、挿入する[3]	✓ はい	✓ はい	✓ はい	✗ いいえ
キー管理機能[4]	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ
暗号化、復号する	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ
署名、検証する	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ
ダイジェストと HMAC の生成	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ

- [1] 基本情報には、クラスター内の HSM 数、各 HSM の IP アドレス、モデル、シリアル番号、デバイス ID、ファームウェア ID などが含まれます。
- [2] ユーザーは、HSM のキーに対応するダイジェスト (ハッシュ) のセットを取得できます。アプリケーションは、これらのダイジェストのセットを比較して、クラスター内の HSM の同期状態を把握します。
- [3] マスクされたオブジェクトは、HSM を離れる前に暗号化されるキーです。これらのオブジェクトを HSM の外部で復号することはできません。これらは、抽出された HSM と同じクラスターにある HSM に挿入された後にのみ復号されます。アプリケーションはマスクされたオブジェクトを抽出して挿入し、クラスター内の HSM を同期します。
- [4] キー管理機能には、キーの属性の作成、削除、ラップ、ラップ解除、変更が含まれます。

CloudHSM CLI で HSM ユーザー管理者を作成する

CloudHSM CLI でハードウェアセキュリティモジュール (CloudHSM) 管理者ユーザーを作成するには、次の手順に従います。

1. CloudHSM CLI インタラクティブモードを起動するには、以下のコマンドを使用します。

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. login コマンドを使用して、管理者としてクラスターにログインします。

```
aws-cloudhsm > login --username <username> --role admin
```

3. システムからパスワードの入力を求められます。パスワードを入力すると、出力はコマンドが成功したことを表示します。

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "admin"
  }
}
```

4. 次のコマンドを入力して管理者を作成します。

```
aws-cloudhsm > user create --username <username> --role admin
```

5. 新しいユーザーのパスワードを入力します。
6. パスワードを再入力して、入力したパスワードが正しいことを確認します。

CloudHSM CLI で HSM Crypto User を作成する

CloudHSM CLI でハードウェアセキュリティモジュール (HSM) Crypto User (CU) を作成するには、次の手順に従います。

1. CloudHSM CLI インタラクティブモードを起動するには、以下のコマンドを使用します。

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. login コマンドを使用して、管理者としてクラスターにログインします。

```
aws-cloudhsm > login --username <username> --role admin
```

3. システムからパスワードの入力を求められます。パスワードを入力すると、出力はコマンドが成功したことを表示します。

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "admin"
  }
}
```

4. 以下のコマンドを入力して、Crypto User を作成します。

```
aws-cloudhsm > user create --username <username> --role crypto-user
```

5. 新しい Crypto User のパスワードを入力します。
6. パスワードを再入力して、入力したパスワードが正しいことを確認します。

CloudHSM CLI でクラスター内のすべての HSM ユーザーを一覧表示する

CloudHSM CLI で user list コマンドを使って、 AWS CloudHSM クラスター上のすべてのユーザーを一覧表示します。user list を実行するのに、ログインする必要はありません。すべてのユーザータイプでユーザーを一覧表示できます。

クラスター内のすべてのユーザーを一覧表示するには、次の手順に従います

1. CloudHSM CLI インタラクティブモードを起動するには、以下のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. クラスター内のすべてのユーザーを一覧表示するには、以下のコマンドを入力します。

   ```
   aws-cloudhsm > user list
   ```

user list の詳細については、ユーザーリスト を参照してください。

CloudHSM CLI で HSM ユーザーパスワードを変更する

CloudHSM CLI の user change-password コマンドを使って、ハードウェアセキュリティモジュール (HSM) ユーザーのパスワードを変更します。

ユーザータイプとパスワードは大文字と小文字が区別されますが、ユーザー名では区別されません。

管理者、Crypto User (CU)、Appliance User (AU) は、自分のパスワードを変更できます。別のユーザーのパスワードを変更する場合、管理者としてログインする必要があります。ただし、現在ログインしているユーザーのパスワードを変更することはできません。

自分のパスワードの変更

1. CloudHSM CLI インタラクティブモードを起動するには、以下のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. login コマンドを使用して、変更するパスワードを使用するユーザーとしてログインします。

```
aws-cloudhsm > login --username <username> --role <role>
```

3. ユーザーのパスワードを入力します。

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "<role>"
  }
}
```

4. user change-password コマンドを入力します。

```
aws-cloudhsm > user change-password --username <username> --role <role>
```

5. 新しいパスワードを入力します。
6. 新しいパスワードを再入力します。

別のユーザーのパスワードの変更

1. CloudHSM CLI インタラクティブモードを起動するには、以下のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、管理者としてログインします。

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

3. パスワードを変更するユーザーのユーザー名とともに user change-password コマンドを入力します。

```
aws-cloudhsm > user change-password --username <username> --role <role>
```

4. 新しいパスワードを入力します。
5. 新しいパスワードを再入力します。

user change-password については、「user change-password」を参照してください。

CloudHSM CLI で HSM ユーザーを削除する

CloudHSM CLI の user delete を使って、ハードウェアセキュリティモジュール (HSM) ユーザーを削除します。別のユーザーを削除する場合、管理者としてログインする必要があります。

> **Tip**
> キーを所有している Crypto User (CU) を削除することはできません。

ユーザーの削除

1. CloudHSM CLI インタラクティブモードを起動するには、以下のコマンドを使用します。

   Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. login コマンドを使用して、管理者としてクラスターにログインします。

```
aws-cloudhsm > login --username <username> --role admin
```

3. システムからパスワードの入力を求められます。パスワードを入力すると、出力はコマンドが成功したことを表示します。

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "admin"
  }
}
```

4. ユーザーを削除するには、user delete コマンドを使用します。

```
aws-cloudhsm > user delete --username <username> --role <role>
```

user delete の詳細情報は、deleteUser を参照してください。

CloudHSM CLI で HSM ユーザーの MFA を管理する

セキュリティを強化するために、 AWS CloudHSM クラスターの保護に役立つように、ユーザーの多要素認証 (MFA) を設定できます。

MFA が有効なハードウェアセキュリティモジュール (HSM) ユーザーアカウントを使用してクラスターにログインする場合、CloudHSM CLI でパスワード (最初の要素、既知の情報) を入力します。次に、CloudHSM CLI からトークンを取得し、そのトークンに署名するように求められます。

2 番目の要素 (自分の持っているもの) を提供する場合、すでに作成して HSM ユーザーに紐づけしたキーペアからプライベートキーを使用してトークンに署名します。クラスターにアクセスする場合、署名付きトークンを CloudHSM CLI に指定します。

ユーザーの MFA 設定の詳細については、「CloudHSM CLI 用の MFA のセットアップ」を参照してください

次のトピックでは、 AWS CloudHSMでのクォーラム認証の操作について詳しく説明します。

トピック

- CloudHSM CLI を使用した AWS CloudHSM クラスターのクォーラム認証と MFA
- CloudHSM CLI AWS CloudHSM を使用するための MFA キーペアの要件
- CloudHSM CLI 用の MFA のセットアップ
- CloudHSM CLI で MFA が有効になっているユーザーを作成する
- CloudHSM CLI で MFA が有効になっているユーザーにログインする
- CloudHSM CLI に対して MFA が有効になっているユーザーのキーをローテーションする
- CloudHSM CLI を使用して MFA パブリックキーの登録を解除する
- CloudHSM CLI を使用した MFA のトークンファイルリファレンス

CloudHSM CLI を使用した AWS CloudHSM クラスターのクォーラム認証と MFA

AWS CloudHSM クラスターは、クォーラム認証と多要素認証 (MFA) に同じキーを使用します。これは、MFA が有効になっているユーザーが実質的に MofN またはクォーラムアクセス制御に登録されていることを意味します。同じ HSM ユーザーに対して MFA 認証とクォーラム認証を正常に実行する際、次の点を考慮する必要があります。

- 現在、ユーザーに対してクォーラム認証を使用している場合は、クォーラムのユーザーに対して作成したものと同じキーペアを使用し、ユーザーに対して MFA を有効化する必要があります。
- クォーラム認証ユーザーではない非 MFA ユーザーに MFA 要件を追加する場合は、そのユーザーを MFA 認証を備えたクォーラム (MofN) 登録ユーザーとして登録します。
- MFA 要件を削除するか、クォーラム認証ユーザーでもある MFA ユーザーのパスワードを変更する場合、クォーラム (MofN) ユーザーとしてそのユーザーの登録も削除されます。
- MFA 要件を削除するか、クォーラム認証ユーザーでもある MFA ユーザーのパスワードを変更する場合、それでもそのユーザーがクォーラム認証に加わる必要がある場合、当該ユーザーをクォーラム (MofN) ユーザーとして再登録する必要があります。

認証の詳細情報は、「クォーラム認証の管理 (M/N)」を参照してください。

CloudHSM CLI AWS CloudHSM を使用するための MFA キーペアの要件

でハードウェアセキュリティモジュール (HSM) ユーザーの多要素認証 (MFA) を有効にするには AWS CloudHSM、新しいキーペアを作成するか、次の要件を満たす既存のキーを使用できます。

- キータイプ : 非対称
- キーの使用方法 : 署名と検証
- キースペック: RSA_2,048
- 署名アルゴリズムには以下が含まれます。 sha256WithRSAEncryption

> **Note**
>
> クォーラム認証を使用している場合、またはクォーラム認証を使用する予定の場合は、「CloudHSM CLI を使用した AWS CloudHSM クラスターのクォーラム認証と MFA」を参照してください

CloudHSM CLI とキーペアを使用して、MFA を有効化した新しい管理者ユーザーを作成できます。

CloudHSM CLI 用の MFA のセットアップ

CloudHSM CLI の多要素認証 (MFA) を設定するには、次の手順に従います。

1. トークン署名戦略を使用して MFA をセットアップするには、まず 2,048 ビットの RSA プライベートキーと関連するパブリックキーを生成する必要があります。

```
$ openssl genrsa -out officer1.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................+++++
.....................................................................+++++
e is 65537 (0x010001)

$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
writing RSA key
```

2. 次のコマンドを使用して CLI をインタラクティブモードで起動します。

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

3. CloudHSM CLI を使用して、ユーザーアカウントにログインします。

```
aws-cloudhsm > login --username <admin> --role <admin> --cluster-id <cluster ID>
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "<admin>"
  }
}
```

4. 次に、コマンドを実行して MFA ストラテジーを変更します。パラメーター --token を指定する必要があります。このパラメータは、署名されていないトークンが書き込まれるファイルを指定します。

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <username> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

5. これで、署名が必要な未署名のトークンを含むファイルが作成されました: unsigned-tokens.json。このファイル内のトークンの数は、クラスター内の HSM の数によって異なります。各トークンは 1 つの HSM を表します。このファイルは JSON 形式で、プライベートキーを持っていることを証明するための署名が必要なトークンが含まれています。

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
{
    {
```

```
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

6. 次のステップは、ステップ 1 で作成したプライベートキーを使用してこれらのトークンに署名することです。署名をファイルに戻します。まず、base64 でエンコードされたトークンを抽出してデコードする必要があります。

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

7. これで、ステップ 1 で作成した RSA プライベートキーを使用して署名できるバイナリトークンができました。

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
```

```
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

8. これで、トークンのバイナリ署名ができました。base64 を使用してエンコードし、トークンファイルに戻す必要があります。

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

9. 最後に、base64 値をコピーしてトークンファイルに貼り付けます。

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

10. トークンファイルに必要な署名がすべて揃ったので、次に進むことができます。署名されたトークンを含むファイルの名前を入力し、Enter キーを押します。最後に、パブリックキーのパスを入力します。

```
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "crypto-user"
  }
}
```

これで、MFA を使用してユーザーを設定できました。

```
{
    "username": "<username>",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
},
```

CloudHSM CLI で MFA が有効になっているユーザーを作成する

多要素認証 (MFA) が有効になっている AWS CloudHSM ユーザーを作成するには、次の手順に従います。

1. CloudHSM CLI を使用し、管理者として HSM にログインします。
2. user create コマンドを使用して、任意のユーザーを作成します。次に、CloudHSM CLI 用の MFA のセットアップ の手順に従ってユーザーの MFA を設定します。

CloudHSM CLI で MFA が有効になっているユーザーにログインする

多要素認証 (MFA) が有効になっている AWS CloudHSM ユーザーにログインするには、次の手順に従います。

1. CloudHSM CLI の login mfa-token-sign コマンドを使用して、MFA が有効になっているユーザーのログインプロセスを MFA で開始します。

```
aws-cloudhsm > login --username <username> --role <role> mfa-token-sign --
token <unsigned-tokens.json>
Enter password:
```

2. パスワードを入力します。次に、署名されていないトークンと署名されたトークンのペアを含むトークンファイルへのパスを入力するよう求められます。ここで、署名付きトークンは、プライベートキーを使用して生成されたものです。

```
aws-cloudhsm > login --username <username> --role <role> mfa-token-sign --
token <unsigned-tokens.json>
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
```

3. 署名済みトークンのファイルパスを入力するように求められますが、別の端末で署名されていないトークンファイルを調べることができます。署名が必要な未署名のトークンを含むファイルを特定します: *<unsigned-tokens.json>*。このファイル内のトークンの数は、クラスター内の HSM の数によって異なります。各トークンは 1 つの HSM を表します。このファイルは JSON 形式で、プライベートキーを持っていることを証明するための署名が必要なトークンが含まれています。

```
$ cat <unsigned-tokens.json>
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
```

```
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. ステップ 2 で作成したプライベートキーを使用して、署名されていないトークンに署名します。まず、base64 でエンコードされたトークンを抽出してデコードする必要があります。

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

5. これで、バイナリトークンが作成されました。MFA セットアップのステップ 1 で作成した RSA プライベートキーを使用して署名します。

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

6. これで、トークンのバイナリ署名ができました。base64 を使用してエンコードし、トークンファイルに戻します。

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
```

```
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. 最後に、base64 値をコピーしてトークンファイルに貼り付けます。

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

8. トークンファイルに必要な署名がすべて揃ったので、次に進むことができます。署名されたトークンを含むファイルの名前を入力し、Enter キーを押します。これで、正常に口グインできたはずです。

```
aws-cloudhsm > login --username <username> --role <role> mfa-token-sign --
token <unsigned-tokens.json>
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
```

```
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "<role>"
  }
}
```

CloudHSM CLI に対して MFA が有効になっているユーザーのキーをローテーションする

多要素認証 (MFA) が有効になっている AWS CloudHSM ユーザーのキーを更新するには、次の手順に従います。

<result>

生成された JSON 形式のトークンファイルにプライベートキーで署名し、新しい MFA パブリックキーを登録しました。
</result>

1. CloudHSM CLI を使用して、任意の管理者または MFA が有効になっている特定のユーザーとして HSM にログインします (詳細については、「MFA が有効になっているユーザーのログイン」を参照)。
2. 次に、コマンドを実行して MFA ストラテジーを変更します。パラメーター --token を指定する必要があります。このパラメータは、署名されていないトークンが書き込まれるファイルを指定します。

   ```
   aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
   username <username> --role crypto-user --change-quorum
   Enter password:
   Confirm password:
   ```

3. 署名が必要な未署名のトークンを含むファイルを特定します: unsigned-tokens.json。このファイル内のトークンの数は、クラスター内の HSM の数によって異なります。各トークンは 1 つの HSM を表します。このファイルは JSON 形式で、プライベートキーを持っていることを証明するための署名が必要なトークンが含まれています。これは、現在登録されているパブリックキーのローテーションに使用したい新しい RSA パブリック/プライベートキーペアからの新しいプライベートキーになります。

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. これらのトークンには、セットアップ時に作成したプライベートキーで署名します。まず、base64 でエンコードされたトークンを抽出してデコードする必要があります。

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

5. これで、バイナリトークンが作成されました。セットアップ時に作成した RSA プライベートキーを使用して署名します。

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
```

```
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

6. これで、トークンのバイナリ署名ができました。base64 を使用してエンコードし、トークンファイルに戻します。

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. 最後に、base64 値をコピーしてトークンファイルに貼り付けます。

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
```

```
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

8. トークンファイルに必要な署名がすべて揃ったので、次に進むことができます。署名されたトークンを含むファイルの名前を入力し、Enter キーを押します。最後に、新しいパブリックキーのパスを入力します。これで、user listの出力の一部として、次の内容が表示されます。

```
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "crypto-user"
  }
}
```

これで、ユーザーに MFA を設定できました。

```
{
    "username": "<username>",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
},
```

CloudHSM CLI を使用して MFA パブリックキーの登録を解除する

MFA パブリックキーが登録されているときに、 AWS CloudHSM 管理者ユーザーの多要素認証 (MFA) パブリックキーを登録解除するには、次の手順に従います。

1. CloudHSM CLI を使用し、MFA を有効化した管理者として HSM にログインします。

2. user change-mfa token-sign コマンドを使用して、ユーザーの MFA を削除します。

```
aws-cloudhsm > user change-mfa token-sign --username <username> --role admin --
deregister --change-quorum
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "admin"
  }
}
```

CloudHSM CLI を使用した MFA のトークンファイルリファレンス

多要素認証 (MFA) パブリックキーを登録するとき、または MFA を使用して CloudHSM CLI にログインしようとするときに生成されるトークンファイルには、次の内容が含まれます。

- トークン: JSON オブジェクトリテラルの形式で、base64 でエンコードされた署名なし/署名付きトークンのペアの配列。
- 署名なし: Base64 でエンコードされ、SHA256 ハッシュされたトークン。
- 署名済み: RSA 2,048 ビットプライベートを使用した、署名されていないトークンの base64 でエンコードされた署名付きトークン (署名)。

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/TK0PVaxLN42X
+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37+j/
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
```

```
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQdErI
    }
  ]
}
```


CloudHSM CLI を使用したクォーラム認証の管理 (M of N アクセスコントロール)

AWS CloudHSM クラスターは、M of N アクセスコントロールとも呼ばれるクォーラム認証をサポートしています。この機能では、特定のオペレーションを実行するために HSM ユーザーが協力する必要があり、追加の保護層が提供されます。

クォーラム認証を使用すると、HSM 上の単一のユーザーがクォーラム管理対象のオペレーションを実行することはできません。代わりに、HSM ユーザーの最小数 (少なくとも 2 人) が、これらのオペレーションを協力して行う必要があります。

クォーラム認証は次のオペレーションを制御できます。

- 管理者による HSM ユーザーの管理: HSM ユーザーの作成と削除または別の HSM ユーザーのパスワードの変更。詳細については、「CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたユーザー管理」を参照してください。

AWS CloudHSMにおけるクォーラム認証の主なポイントは次のとおりです。

- HSM ユーザーは自身のクォーラムトークンに署名できます。つまり、クォーラム認証に必要な承認の 1 つを自分で提供できます。
- クォーラム承認者の最小数は、2 ～ 8 の範囲で選択します。

- HSM はクォーラムトークンを最大 1,024 保存できます。この上限に達すると、有効期限切れのトークンが削除され、新しいトークンが作成されます。
- トークンは、デフォルトでは作成後 10 分で有効期限が切れます。
- MFA が有効になっているクラスターでは、クォーラム認証と多要素認証 (MFA) に同じキーが使用されます。詳細については、CloudHSM CLI を使用して MFA を管理するを参照してください。
- 各 HSM の管理者サービスごとに存在できるクォーラムトークンは 1 つだけです。同じサービスに対して新しいクォーラムトークンを生成する前に、サービスのアクティブなクォーラムトークンを完了または削除する必要があります。詳細については、「サポートされているサービスとタイプ」を参照してください。

次のトピックでは、 AWS CloudHSMでのクォーラム認証についてさらに詳細な情報を提供します。

トピック

- CloudHSM CLI のクォーラム認証プロセス
- CloudHSM CLI によるクォーラム認証でサポートされている AWS CloudHSM サービス名とタイプ
- CloudHSM CLI を使用して AWS CloudHSM 管理者のクォーラム認証を設定する
- CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたユーザー管理
- CloudHSM CLI AWS CloudHSM を使用するためのクォーラム最小値を変更する

CloudHSM CLI のクォーラム認証プロセス

次の手順は、CloudHSM CLI のクォーラム認証プロセスの概要を示しています。特定のステップとツールについては、CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたユーザー管理 を参照してください。

1. 各ハードウェアセキュリティモジュール (HSM) ユーザーは、署名のための非対称キーを作成します。これは HSM の外部で行い、キーを適切に保護します。
2. 各 HSM ユーザーは HSM にログインし、署名キーの公開部分 (パブリックキー) を HSM に登録します。
3. HSM ユーザーがクォーラム管理されたオペレーションを実行する場合は、HSM にログインし、クォーラムトークンを取得します。
4. HSM ユーザーは、クォーラムトークンを 1 人または複数の他の HSM ユーザーに付与し、承認を求めます。

5. 他の HSM ユーザーは、キーを使用してクォーラムトークンに暗号で署名することにより承認します。これは HSM の外部で行われます。
6. HSM ユーザーが必要な数の承認を得たら、同じユーザーが HSM にログインし、必要な承認 (署名) をすべて含む署名付きクォーラムトークンファイルを提供して、--approval 引数を指定してクォーラム制御オペレーションを実行します。
7. HSM では、それぞれの署名した人の登録されたパブリックキーを使用して署名を確認します。署名が有効な場合、HSM はトークンを承認し、クォーラム制御されたオペレーションが実行されます。

CloudHSM CLI によるクォーラム認証でサポートされている AWS CloudHSM サービス名とタイプ

管理サービス: クォーラム認証は、ユーザーの作成、ユーザーの削除、ユーザーパスワードの変更、クォーラム値の設定、クォーラム機能と MFA 機能の無効化などの管理者権限を持つサービスに使用されます。

Crypto User サービス: クォーラム認証は、特定のキーに関連する Crypt User の特権サービス (キーによる署名、キーの共有/共有解除)、キーのラッピング/ラップ解除、キー属性の設定など) に使用されます。関連付けられたキーのクォーラム値は、そのキーを生成、インポート、またはラップ解除する際に設定されます。クォーラム値は、そのキーが関連付けられているユーザー数 (キーが共有されているユーザーおよびキー所有者を含む) 以下である必要があります。

各サービスタイプはさらに適格なサービス名に分類されます。このサービス名には、実行可能なクォーラムがサポートする特定のサービスオペレーションのセットが含まれます。

サービス名	サービスタイプ	サービスオペレーション
ユーザー	管理者	• user create • user delete • user change-password • user change-mfa
quorum	管理者	• quorum token-sign set-quorum-value
cluster[1]	管理者	• cluster mtls register-trust-anchor

サービス名	サービスタイプ	サービスオペレーション
		• cluster mtls deregister-trust-anchor • cluster mtls set-enforcement
キーの管理	Crypto User	• キーのラップ • キーのラップ解除 • キーシェア • キー共有解除 • key set-attribute
キーの用途	Crypto User	• キーサイン

[1] クラスターサービスは hsm2m.medium でのみ利用できます

CloudHSM CLI を使用して AWS CloudHSM 管理者のクォーラム認証を設定する

以下のトピックでは、 AWS CloudHSM 管理者がクォーラム認証を使用できるようにハードウェアセキュリティモジュール (HSM) を設定するために完了する必要がある手順について説明します。管理者のクォーラム認証を最初に設定する場合に、これらのステップを 1 回だけ実行する必要があります。これらのステップが完了したら、CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたユーザー管理 を参照してください。

トピック

- 前提条件
- ステップ 1. 署名のためのキーの作成と登録
- ステップ 2. HSM のクォーラム最小値を設定する
- クォーラム最小値

前提条件

この例を理解するには、CloudHSM CLI についての知識が必要です。

ステップ 1. 署名のためのキーの作成と登録

クォーラム認証を使用する場合、各管理者が以下のすべてのステップを実行する必要があります。

トピック

- RSA キーペアの作成
- 登録トークンの作成と署名
- HSM でパブリックキーを登録する

RSA キーペアの作成

様々なキーペアを作成、保護する方法があります。次の例では、OpenSSL 使用方法を説明しています。

Example— OpenSSL でプライベートキーを作成する

次の例は、OpenSSL を使用して 2,048 ビットの RSA キーを作成する方法を示しています。この例を使用するには、*<admin.key>* を、キーの保存先のファイル名に置き換えてください。

```
$ openssl genrsa -out <admin.key>
Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
```

次に、作成したプライベートキーを使用してパブリックキーを生成します。

Example— OpenSSL でパブリックキーを作成する

以下の例は、OpenSSL を使用して先ほど作成したプライベートキーからパブリックキーを作成する方法を示しています。

```
$ openssl rsa -in admin.key -outform PEM -pubout -out admin1.pub
writing RSA key
```

登録トークンの作成と署名

トークンを作成し、前のステップで生成したプライベートキーを使用して署名します。

Example– 登録トークンの作成と署名

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. quorum token-sign generate コマンドを実行して登録トークンを作成します。

```
aws-cloudhsm > quorum token-sign generate --service registration --token /path/
tokenfile
{
  "error_code": 0,
  "data": {
    "path": "/path/tokenfile"
  }
}
```

3. quorum token-sign generate コマンドは、指定されたファイルパスに登録トークンを生成します。トークンファイルを調査します。

```
$ cat /path/tokenfile
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

トークンファイルは、次のもので構成されます。

- approval_data: base64 でエンコードされランダム化されたデータトークン。raw データが最大 245 バイトを超えないもの。
- unsigned: base64 でエンコードされ、SHA256 ハッシュされた approval_data のトークン。

- signed: OpenSSL で以前に生成された RSA 2,048 ビットのプライベートキーを使用した、署名されていないトークンの base64 でエンコードされた署名付きトークン (署名)。

プライベートキーを使用して署名なしトークンに署名し、プライベートキーへのアクセス権があることを示します。管理者をクォーラムユーザーとして AWS CloudHSM クラスターに登録するには、登録トークンファイルに署名とパブリックキーが完全に入力されている必要があります。

Example– 署名なし登録トークンへ署名する

1. base64 でエンコードされた署名なしトークンをデコードし、バイナリファイルに入れます。

```
$ echo -n '6BMUj6mUjjko6ZLCEdzGlWpR5sILhFJfqhW1ej3Oq1g=' | base64 -d > admin.bin
```

2. OpenSSL とプライベートキーを使用して現在の署名なしバイナリ登録トークンに署名し、バイナリ署名ファイルを作成します。

```
$ openssl pkeyutl -sign \
-inkey admin.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. バイナリ署名を base64 にエンコードします。

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

4. base64 でエンコードされた署名をコピーしてトークンファイルに貼り付けます。

```
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": <signed token in base64 encoding>
    }
  ]
}
```

HSM でパブリックキーを登録する

キーを作成した後、管理者はパブリックキーを AWS CloudHSM クラスターに登録する必要があります。

HSM にパブリックキーの登録するには

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、管理者としてログインします。

   ```
   aws-cloudhsm > login --username <admin> --role admin
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "<admin>",
       "role": "admin"
     }
   }
   ```

3. CloudHSM CLI を使用してユーザーのトークン署名クォーラム戦略を登録する コマンドを使用してパブリックキーを登録します。詳細については、次の例を参照するか、または help user change-quorum token-sign register コマンドを使用してください。

Example– AWS CloudHSM クラスターにパブリックキーを登録する

以下の例では、CloudHSM CLI で user change-quorum token-sign register コマンドを使用して、管理者のパブリックキーを HSM に登録する方法を示しています。このコマンドを使用するには、管理者が HSM にログインしている必要があります。以下の値を自分の値に置き換えてください。

```
aws-cloudhsm > user change-quorum token-sign register --public-key </path/admin.pub> --
signed-token </path/tokenfile>
```

```
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

> **Note**
>
> /path/admin.pub: パブリックキー PEM ファイルへのファイルパス
> 必須: はい
> /path/tokenfile: ユーザーのプライベートキーによって署名されたトークンを含むファイルパス
> 必須: はい

すべての管理者がパブリックキーを登録すると、user list コマンドの出力のクォーラムフィールドには次のように表示され、有効になっているクォーラム戦略が使用中であることが示されます。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
```

```
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
```

```
  }
}
```

この例では、 user list コマンドからの次の出力に示すように、 AWS CloudHSM クラスターには 2 つの HSMs があり、それぞれが同じ管理者を持ちます。ユーザー作成の詳細については、CloudHSM CLI によるユーザー管理 を参照してください。

ステップ 2. HSM のクォーラム最小値を設定する

クォーラム認証を使用するには、管理者が HSM にログインしてクォーラム最小値を設定する必要があります。これは、HSM ユーザー管理オペレーションを実行するために必要な管理者承認の最小数です。HSM 上の任意の管理者は、署名用のキーを登録していない管理者を含むクォーラム最小値を設定できます。クォーラム最小値はいつでも変更できます。詳細については、「最小値を変更」を参照してください。

HSM のクォーラム最小値の設定

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、管理者としてログインします。

   ```
   aws-cloudhsm > login --username <admin> --role admin
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "<admin>",
       "role": "admin"
     }
   }
   ```

3. クォーラム最小値を設定する場合、CloudHSM CLI でクォーラム値を更新する コマンドを使用します。`--service` フラグは、値を設定する対象の HSM サービスを指定します。詳細につい

ては、次の例を参照するか、または help quorum token-sign set-quorum-value コマンドを使用してください。

Example HSM のクォーラム最小値を設定する

この例では、クォーラム最小値 2 を使用します。最大は HSM 上の管理者の合計数で、2 から 8 までの任意の値を選択できます。この例では、HSM には 4 人の管理者がいるため、設定可能な最大値は 4 です。

次のコマンド例を使用するには、最後の数値 (*<2>*) を所望のフォーラム最小値に置き換えてください。

```
aws-cloudhsm > quorum token-sign set-quorum-value --service user --value <2>
{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

この例では、CloudHSM CLI でクォーラム値を表示する コマンドは、サービスに含まれる HSM サービスのタイプ、名前、および説明を示しています。

クォーラム最小値

サービスのクォーラム最小値を取得するには、quorum token-sign list-quorum-values コマンドを使用します。

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

前述の quorum token-sign list-quorum-values コマンドの出力は、ユーザー管理オペレーションを担当する HSM ユーザーサービスのクォーラム最小値が 2 になったことを示しています。これらのステップが完了したら、クォーラムによるユーザー管理 (M of N) を参照してください。

管理サービス: クォーラム認証は、ユーザーの作成、ユーザーの削除、ユーザーパスワードの変更、クォーラム値の設定、クォーラム機能と MFA 機能の無効化などの管理者権限を持つサービスに使用されます。

Crypto User サービス: クォーラム認証は、特定のキーに関連する Crypt User の特権サービス (キーによる署名、キーの共有/共有解除)、キーのラッピング/ラップ解除、キー属性の設定など) に使用されます。関連付けられたキーのクォーラム値は、そのキーを生成、インポート、またはラップ解除する際に設定されます。クォーラム値は、そのキーが関連付けられているユーザー数 (キーが共有されているユーザーおよびキー所有者を含む) 以下である必要があります。

各サービスタイプはさらに適格なサービス名に分類されます。このサービス名には、実行可能なクォーラムがサポートする特定のサービスオペレーションのセットが含まれます。

サービス名	サービスタイプ	サービスオペレーション
ユーザー	管理者	• user create • user delete • user change-password • user change-mfa
quorum	管理者	• quorum token-sign set-quorum-value
cluster[1]	管理者	• cluster mtls register-trust-anchor • cluster mtls deregister-trust-anchor • cluster mtls set-enforcement
キーの管理	Crypto User	• キーのラップ • キーのラップ解除 • キーシェア • キー共有解除 • key set-attribute
キーの用途	Crypto User	• キーサイン

[1] クラスターサービスは hsm2m.medium でのみ利用できます

CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたユーザー管理

ハードウェアセキュリティモジュール (HSM) AWS CloudHSM ???の管理者は、 AWS CloudHSM クラスター内の次のオペレーションのクォーラム認証を設定できます。

- CloudHSM CLI を使用して AWS CloudHSM ユーザーを作成する
- CloudHSM CLI で AWS CloudHSM ユーザーを削除する
- CloudHSM CLI でユーザーのパスワードを変更する
- CloudHSM CLI のユーザー change-mfa カテゴリ

AWS CloudHSM クラスターがクォーラム認証用に設定されている場合、管理者は HSM ユーザー管理オペレーションを自分で実行することはできません。次の例は、管理者が HSM で新しいユーザーを作成しようとしたときの出力を示しています。コマンドは失敗し、クォーラム認証が必要であることを示すエラーが表示されます。

```
aws-cloudhsm > user create --username user1 --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

HSM ユーザー管理オペレーションを実行するには、管理者は以下のタスクを完了する必要があります。

トピック

- ステップ 1. クォーラムトークンの取得
- ステップ 2. 承認する管理者から署名を取得する
- ステップ 3. AWS CloudHSM クラスターでトークンを承認し、ユーザー管理オペレーションを実行する

ステップ 1. クォーラムトークンの取得

まず、管理者は CloudHSM CLI を使用してクォーラムトークンをリクエストする必要があります。

クォーラムトークンを取得するには

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、管理者としてログインします。

   ```
   aws-cloudhsm > login --username <admin> --role admin
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "<admin>",
       "role": "admin"
     }
   }
   ```

3. quorum token-sign generate コマンドを使用してクォーラムトークンを生成します。詳細については、次の例を参照するか、または help quorum token-sign generate コマンドを使用してください。

Example– クォーラムトークンを生成する

この例では、ユーザー名を admin とする管理者のクォーラムトークンを取得し、そのトークンを admin.token というファイルに保存します。この例のコマンドを使用するには、以下の値を独自のものに置き換えてください。

- *<admin>* – トークンを取得する管理者の名前。HSM にログインし、このコマンドを実行している管理者と同じであることが必要です。
- *<admin.token>* – クォーラムトークンを保存するファイルの名前。

次のコマンドで、user は生成するトークンを使用できるサービス名を識別します。この場合、トークンは HSM ユーザー管理オペレーション (user サービス) 用です。

```
aws-cloudhsm > login --username <admin> --role admin --password <password>
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

```
aws-cloudhsm > quorum token-sign generate --service user --token </path/admin.token>
{
  "error_code": 0,
  "data": {
    "path": "/home/tfile"
  }
}
```

quorum token-sign generate コマンドは、指定されたファイルパスでユーザーサービスのクォーラムトークンを生成します。トークンファイルは以下のように検査できます。

```
$ cat </path/admin.token>
{
  "version": "2.0",
  "service": "user-management",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

トークンファイルは、次のもので構成されます。

- service: トークンが関連付けられているクォーラムサービスの識別子。
- approval_data: HSM によって生成された base64 でエンコードされた raw データトークン。
- token: base64 でエンコードされ、SHA-256 ハッシュされた approval_data のトークン

- signatures: base64 でエンコードされた署名なしトークンの署名済みトークン (署名) の配列。承認者の各署名は JSON オブジェクトリテラルの形式になっています。

```
{
      "username": "<APPROVER_USERNAME>",
      "role": "<APPROVER_ROLE>",
      "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
}
```

  各署名は、パブリックキーが HSM に登録されている承認者が、対応する RSA 2,048 ビットプライベートキーを使用した結果から作成されます。

生成されたユーザーサービスのクォーラムトークンが CloudHSM クラスターに存在していることは、quorum token-sign list コマンドを実行することによって確認できます。

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "admin",
        "service": "user",
        "approvals-required": {
          "value": 2
        },
        "number-of-approvals": {
          "value": 0
        },
        "token-timeout-seconds": {
          "value": 597
        },
        "cluster-coverage": "full"
      }
    ]
  }
}
```

`token-timeout-seconds` 時間は、生成されたトークンの有効期限が切れる前に承認されるまでのタイムアウト時間を秒単位で示します。

ステップ 2. 承認する管理者から署名を取得する

クォーラムトークンを持つ管理者は、そのトークンを他の管理者に承認してもらう必要があります。他の管理者は、承認を与えるために、署名キーを使用してトークンを暗号で署名します。この署名は HSM 外で行われます。

トークンの署名にはさまざまな方法が使用されます。次の例では、OpenSSL を使用しています。別の署名ツールを使用する場合は、そのツールで必ず管理者のプライベートキー (署名キー) を使用してトークンの SHA-256 ダイジェストに署名します。

Example– 承認する管理者から署名を取得する

この例では、トークン (admin) を持つ管理者に少なくとも 2 つの承認が必要です。以下のコマンド例では、2 人の管理者が OpenSSL を使用してトークンに暗号で署名する方法を示します。

1. base64 でエンコードされた署名なしトークンをデコードし、バイナリファイルに入れます。

```
$ echo -n '0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=' | base64 -d > admin.bin
```

2. OpenSSL と、承認者 (admin3) のそれぞれのプライベートキーを使用して、ユーザーサービスの現在のバイナリクォーラム署名なしトークンに署名し、バイナリ署名ファイルを作成します。

```
$ openssl pkeyutl -sign \
-inkey admin3.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. バイナリ署名を base64 にエンコードします。

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

4. 最後に、承認者署名用に以前に指定した JSON オブジェクトリテラル形式に従って、base64 でエンコードされた署名をコピーしてトークンファイルに貼り付けます。

```
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAA
+b23gAAAAAAAAAA",
```

```
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": [
    {
      "username": "admin2",
      "role": "admin",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    },
    {
      "username": "admin3",
      "role": "admin",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw==
    }
  ]
}
```


ステップ 3. AWS CloudHSM クラスターでトークンを承認し、ユーザー管理オペレーションを実行する

前のセクションで説明したように、管理者は必要な承認/署名を取得すると、管理者は以下のユーザー管理オペレーションのいずれかとともにそのトークンを AWS CloudHSM クラスターに提供することができます。

- 作成
- 削除
- change-password
- user change-mfa

これらのコマンドの詳しい使用方法については、CloudHSM CLI によるユーザー管理 を参照してください。

トランザクション中、トークンは AWS CloudHSM クラスター内で承認され、リクエストされたユーザー管理オペレーションを実行します。ユーザー管理オペレーションが成功するかどうかは、承認された有効なクォーラムトークンと有効なユーザー管理オペレーションの両方に左右されます。

管理者は、トークンを 1 つのオペレーションにのみ使用できます。そのオペレーションが成功すると、トークンは無効になります。別の HSM ユーザー管理オペレーションを行うには、管理者は上記の手順を繰り返す必要があります。つまり、管理者は新しいクォーラムトークンを取得し、承認者から新しい署名を取得した後で、要求されたユーザー管理オペレーションによって HSM で新しいトークンを承認して使用する必要があります。

> **Note**
>
> クォーラムトークンは、現在のログインセッションが開いている間だけ有効です。CloudHSM CLI からログアウトするか、ネットワークが切断された場合、トークンは無効になります。同様に、承認されたトークンは CloudHSM CLI 内でのみ使用できます。別のアプリケーションでの認証には使用できません。

Example管理者として新規ユーザーを作成する

次の例では、ログイン中の管理者が HSM で新しいユーザーを作成しています。

```
aws-cloudhsm > user create --username user1 --role crypto-user --approval /path/
admin.token
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "user1",
    "role": "crypto-user"
  }
}
```

その後、管理者は新しいユーザーの作成を確認する user list コマンドを入力します。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
```

```
    {
      "username": "admin",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "admin2",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "admin3",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "admin4",
      "role": "admin",
      "locked": "false",
      "mfa": [],
```

```
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "user1",
        "role": "crypto-user",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

管理者が別の HSM ユーザー管理オペレーションを実行しようとするとクォーラム認証エラーが発生して失敗します。

```
aws-cloudhsm > user delete --username user1 --role crypto-user
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

以下に示すように、quorum token-sign list コマンドは管理者に承認されたトークンがないことを示しています。別の HSM ユーザー管理オペレーションを実行するには、管理者は新しいクォーラムトークンを生成し、承認者から新しい署名を取得し、--approval 引数を使用して目的のユーザー管理オペレーションを実行して、ユーザー管理オペレーションの実行中に承認され使用されるクォーラムトークンを供給する必要があります。

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}
```

CloudHSM CLI AWS CloudHSM を使用するためのクォーラム最小値を変更する

CloudHSM 管理者のクォーラム最小値を設定した後、クォーラム最小値を調整する必要が生じる場合があります。HSM は、承認者の数が現在の値以上である場合にのみ、クォーラム最小値の変更を許可します。たとえば、クォーラム最小値が 2 の場合、少なくとも 2 人の管理者が変更を承認する必要があります。

> **Note**
>
> ユーザーサービスのクォーラム値は、常にクォーラムサービスのクォーラム値以下でなければなりません。サービス名の詳細については、CloudHSM CLI によるクォーラム認証でサポートされている AWS CloudHSM サービス名とタイプ を参照してください。

クォーラム最小値の変更のためにクォーラムの承認を取得する場合、quorum token-sign set-quorum-value コマンドを使用する quorum service のためのクォーラムトークンが必要です。quorum token-sign set-quorum-value コマンドを使用して quorum service のクォーラムトークンを生成するには、クォーラムサービスが 1 より大きい必要があります。つまり、ユーザーサービスのクォーラム最小値を変更するには、クォーラムサービスのクォーラム最小値の変更が必要になる場合があります。

管理者のクォーラム最小値を変更する手順

1. CloudHSM CLI のインタラクティブモードを開始します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、管理者として ログインします。

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

3. 現在のクォーラム最小値を確認します。

```
aws-cloudhsm > quorum token-sign list-quorum-values
```

4. クォーラムサービスのクォーラム最小値がユーザーサービスの値より小さい場合、クォーラムサービスの値を変更します。

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value <3>
```

5. クォーラムサービスのクォーラムトークンを生成します。
6. 他の管理者からの承認 (署名) の取得。
7. CloudHSM クラスター上のトークンを承認し、ユーザー管理オペレーションを実行します。
8. ユーザーサービスのクォーラム最小値を変更します。

```
aws-cloudhsm > quorum token-sign set-quorum-value
```

Exampleクォーラムサービスの最小値を調整する

1. 現在の値を確認します。例では、ユーザーサービスのクォーラム最小値が現在 2 であることを示しています。

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
```

```
}
```

2. クォーラムサービス値を変更します。クォーラムサービスのクォーラム最小値をユーザーサービスの値と同じかそれより大きい値に設定します。この例では、クォーラムサービスのクォーラム最小値を、前の例のユーザーサービスに設定されているのと同じ値である 2 に設定します。

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

3. 変更を確認します。この例は、ユーザーサービスとクォーラムサービスでクォーラム最小値が現在 2 であることを示しています。

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 2
  }
}
```

CloudHSM 管理ユーティリティ (CMU) による HSM ユーザー管理

でハードウェアセキュリティモジュール (HSM) ユーザーを管理するには AWS CloudHSM、Cryptographic Officer (CO) のユーザー名とパスワードを使用して HSM にログインする必要があります。CO のみユーザーを管理できます。HSM には、admin という名前のデフォルト CO が含まれています。admin クラスターのアクティブ化 の際に必要なパスワードを設定しました。

このトピックでは、 AWS CloudHSM 管理ユーティリティ (CMU) を使用して HSM ユーザーを管理する手順と詳細について説明します。

トピック

- Management AWS CloudHSM Utility でのユーザー管理の前提条件
- AWS CloudHSM 管理ユーティリティの HSM ユーザータイプ
- AWS CloudHSM 管理ユーティリティの HSM ユーザーアクセス許可テーブル

- AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーを作成する
- AWS CloudHSM 管理ユーティリティを使用してクラスター内のすべての HSM ユーザーを一覧表示する
- AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーパスワードを変更する
- AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーを削除する
- 管理ユーティリティを使用してユーザーの 2FA AWS CloudHSM を管理する
- CloudHSM 管理ユーティリティ (CMU) を使用したクォーラム認証の管理 (M of N アクセスコントロール)

Management AWS CloudHSM Utility でのユーザー管理の前提条件

AWS CloudHSM 管理ユーティリティ (CMU) を使用して でハードウェアセキュリティモジュール (HSM) ユーザーを管理する前に AWS CloudHSM、以下の前提条件を満たす必要があります。以下のセクションでは、 CMU の基本的な使い方について説明します。

セクション

- で HSM の IP アドレスを取得する AWS CloudHSM
- クライアント SDK 3.2.1 以前のバージョンでの CMU の使用
- CloudHSM 管理ユーティリティのダウンロード

で HSM の IP アドレスを取得する AWS CloudHSM

CMU を使用する場合、設定ツールでローカル設定を更新する必要があります。CMU はクラスターへの独自の接続を作成しますが、この接続はクラスターを 認識しません。クラスター情報を追跡するため、CMU はローカル設定ファイルを保持します。これは、毎回 CMU を使用する際、まず、`--cmu` パラメータを指定して 設定 コマンドラインツールを実行し、設定ファイルを更新する必要があります。クライアント SDK 3.2.1 以前のバージョンを使用している場合、`--cmu` とは異なるパラメータを使用する必要があります。詳細については、「the section called “クライアント SDK 3.2.1 以前のバージョンでの CMU の使用”」を参照してください。

`--cmu` パラメータには、クラスター内の HSM の IP アドレスを追加する必要があります。複数の HSM をお持ちの方は、任意の IP アドレスを使用できます。これで確実に CMU がクラスター全体に加えた変更を伝播できます。CMU はローカルファイルを使用してクラスター情報を追跡することに注意してください。特定のホストから CMU を最後に使用後にクラスターが変更されている場

合、該当するホストに保存されているローカル設定ファイルにそれらの変更を追加する必要があります。CMU の使用中は、HSM を追加または削除しないでください。

HSM の IP アドレスを取得するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. AWS リージョンを変更するには、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. クラスターの詳細ページを開くには、クラスターテーブルでクラスター ID を選択します。
4. IP アドレスを取得するには、HSM タブに移動します。IPv4 クラスターの場合は、ENI IPv4 アドレスにリストされているアドレスを選択します。デュアルスタッククラスターの場合は、ENI IPv4 アドレスまたは ENI IPv6 アドレスのいずれかを使用します。

HSM の IP アドレスを取得する (AWS CLI)

- describe-clusters から AWS CLIコマンドを実行して、HSM の IP アドレスを取得します。コマンドの出力では、HSM の IP アドレスは `EniIp` および `EniIpV6` の値です (デュアルスタッククラスターの場合)。

```
$ aws cloudhsmv2 describe-clusters
{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
                    "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733",
...
```

クライアント SDK 3.2.1 以前のバージョンでの CMU の使用

Client SDK 3.3.0 では、 `--cmu`パラメータのサポート AWS CloudHSM が追加されました。これにより、CMU の設定ファイルを更新するプロセスが簡素化されます。クライアント SDK 3.2.1 以前の

バージョンの CMU を使用している場合は、-a と -m パラメータを引き続き使用して設定ファイルを更新する必要があります。パラメータの詳細情報は、設定ツール を参照してください。

CloudHSM 管理ユーティリティのダウンロード

クライアント SDK 5 およびクライアント SDK 3 の使用の有無を問わず、 HSM ユーザー管理タスクで最新バージョンの CMU を使用できます。

CMU のダウンロードおよびインストール

- CMU をダウンロードおよびインストールします。

  Amazon Linux

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-mgmt-util-latest.el6.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el6.x86_64.rpm
  ```

  Amazon Linux 2

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  CentOS 7.8+

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  CentOS 8.3+

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
  ```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

RHEL 7 (7.8+)

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-mgmt-util_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

Windows Server 2012

1. CloudHSM 管理ユーティリティ をダウンロードします。
2. Windowsの管理権限を持つ CMU インストーラ (AWSCloudHSMManagementUtil-latest.msi) を実行します。

Windows Server 2012 R2

1. CloudHSM 管理ユーティリティ をダウンロードします。
2. Windowsの管理権限を持つ CMU インストーラ (AWSCloudHSMManagementUtil-latest.msi) を実行します。

Windows Server 2016

1. CloudHSM 管理ユーティリティ をダウンロードします。
2. Windowsの管理権限を持つ CMU インストーラ (AWSCloudHSMManagementUtil-latest.msi) を実行します。

AWS CloudHSM 管理ユーティリティの HSM ユーザータイプ

ハードウェアセキュリティモジュール (HSM) で実行するほとんどのオペレーションには、 AWS CloudHSM HSM ユーザーの認証情報が必要です。HSM は各 HSM ユーザーを認証し、各 HSM ユーザーに設定されている タイプ により、ユーザーとして HSM で実行できるオペレーションが決定されます。

> **Note**
>
> HSM ユーザーは IAM ユーザーとは異なります。正しい認証情報を持つ IAM ユーザーは、AWS API を介してリソースを操作することで HSM を作成できます。HSM を作成したら、HSM ユーザー認証情報を使用して HSM でのオペレーションを認証する必要があります。

ユーザータイプ

- Precrypto Officer (PRECO)
- Crypto Officer (CO)
- Crypto User (CU)
- Appliance User (AU)

Precrypto Officer (PRECO)

クラウド管理ユーティリティ (CMU) とキー管理ユーティリティ (KMU) のどちらでも、PRECO は AWS CloudHSM クラスター内の最初の HSM にのみ存在する一時的なユーザーです。新しいクラスターの最初の HSM には、このクラスターがアクティブ化されたことがないことを示す PRECO ユーザーが含まれています。クラスターをアクティブ化するには、cloudhsm-cli を実行し、cluster activate コマンドを実行します。HSM にログインし、PRECO のパスワードを変更します。パスワードを変更すると、このユーザーは Crypto Officer (CO) になります。

Crypto Officer (CO)

クラウド管理ユーティリティ (CMU) とキー管理ユーティリティ (KMU) のどちらでも、Crypto Officer (CO) がユーザー管理オペレーションを実行できます。たとえば、ユーザーの作成および削除と、ユーザーパスワードの変更を行うことなどができます。CO ユーザーの詳細情報は、AWS CloudHSM 管理ユーティリティの HSM ユーザーアクセス許可テーブル を参照してください。新しいクラスターをアクティブ化すると、ユーザーは Precrypto Officer (PRECO) から crypto officer (CO) に変更されます。

Crypto User (CU)

Crypto User (CU) は、以下のキー管理および暗号化のオペレーションを行うことができます。

- キー管理 - 暗号化キーの作成、削除、共有、インポート、エクスポートを行います。
- 暗号化オペレーション - 暗号化キーを使用して、暗号化、復号、署名、検証などを行います。

詳細については、「AWS CloudHSM 管理ユーティリティの HSM ユーザーアクセス許可テーブル」を参照してください。

Appliance User (AU)

アプライアンスユーザー (AU) は、クラスターの HSMs でクローン作成および同期オペレーションを実行できます。 は AU AWS CloudHSM を使用して、 AWS CloudHSM クラスター内の HSMs を同期します。AU は が提供するすべての HSMs に存在し AWS CloudHSM、アクセス許可が制限されています。詳細については、「AWS CloudHSM 管理ユーティリティの HSM ユーザーアクセス許可テーブル」を参照してください。

AWS は HSMs に対してオペレーションを実行できません。 はユーザーまたはキーを表示または変更 AWS できず、これらのキーを使用して暗号化オペレーションを実行できません。

AWS CloudHSM 管理ユーティリティの HSM ユーザーアクセス許可テーブル

次の表は、 AWS CloudHSMで操作を実行できる HSM ユーザーまたはセッションの種類ごとに分類されたハードウェアセキュリティモジュール (HSM) オペレーションを示しています。

	Crypto Officer (CO)	Crypto User (CU)	Appliance User (AU)	未認証セッション
基本的なクラスター情報を取得する[1]	✓ はい	✓ はい	✓ はい	✓ はい
自分のパスワードを変更する	✓ はい	✓ はい	✓ はい	該当しない
ユーザーのパスワードを変更する	✓ はい	✗ いいえ	✗ いいえ	✗ いいえ
ユーザーを追加、削除する	✓ はい	✗ いいえ	✗ いいえ	✗ いいえ
同期のステータスを取得する[2]	✓ はい	✓ はい	✓ はい	✗ いいえ
マスクされたオブジェクトを抽出、挿入する[3]	✓ はい	✓ はい	✓ はい	✗ いいえ
キー管理機能[4]	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ
暗号化、復号する	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ
署名、検証する	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ
ダイジェストと HMAC の生成	✗ いいえ	✓ はい	✗ いいえ	✗ いいえ

- [1] 基本情報には、クラスター内の HSM 数、各 HSM の IP アドレス、モデル、シリアル番号、デバイス ID、ファームウェア ID などが含まれます。
- [2] ユーザーは、HSM のキーに対応するダイジェスト (ハッシュ) のセットを取得できます。アプリケーションは、これらのダイジェストのセットを比較して、クラスター内の HSM の同期状態を把握します。
- [3] マスクされたオブジェクトは、HSM を離れる前に暗号化されるキーです。これらのオブジェクトを HSM の外部で復号することはできません。これらは、抽出された HSM と同じクラスターにある HSM に挿入された後にのみ復号されます。アプリケーションはマスクされたオブジェクトを抽出して挿入し、クラスター内の HSM を同期します。
- [4] キー管理機能には、キーの属性の作成、削除、ラップ、ラップ解除、変更が含まれます。

AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーを作成する

AWS CloudHSM 管理ユーティリティ (CMU) createUserで を使用して、ハードウェアセキュリティモジュール (HSM) に新しいユーザーを作成します。ユーザーを作成する場合、CO としてログインする必要があります。

新しい CO ユーザーの作成

1. 設定ツールで CMU 設定を更新します。

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
   ```

2. CMU を開始します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. CO ユーザーとして HSM にログインします。

```
aws-cloudhsm > loginHSM CO admin co12345
```

接続 CMU リストの数が、クラスター内の HSM 数と一致していることを確認します。一致しない場合、ログアウトして最初からやり直してください。

4. createUser を使用して、 パスワードを **example_officer** に設定して **password1** という CO ユーザーを作成します。

```
aws-cloudhsm > createUser CO example_officer password1
```

CMU は、ユーザーの作成オペレーションについてプロンプトを表示します。

```
*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?
```

5. タイプ **y**。

新しい CU ユーザーの作成

1. 設定ツールで CMU 設定を更新します。

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. CMU を開始します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:
   \ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. CO ユーザーとして HSM にログインします。

   ```
   aws-cloudhsm > loginHSM CO admin co12345
   ```

   接続 CMU リストの数が、クラスター内の HSM 数と一致していることを確認します。一致しない場合、ログアウトして最初からやり直してください。

4. createUser を使用して、**example_user** という CU ユーザーをパスワードは **password1** で作成します。

   ```
   aws-cloudhsm > createUser CU example_user password1
   ```

   CMU は、ユーザーの作成オペレーションについてプロンプトを表示します。

   ```
   ************************CAUTION*******************************
   This is a CRITICAL operation, should be done on all nodes in the
   cluster. AWS does NOT synchronize these changes automatically with the
   nodes on which this operation is not executed or failed, please
   ensure this operation is executed on all nodes in the cluster.
   **************************************************************

   Do you want to continue(y/n)?
   ```

5. タイプ **y**。

createUser の詳細情報は、createUser を参照してください。

AWS CloudHSM 管理ユーティリティを使用してクラスター内のすべての HSM ユーザーを一覧表示する

AWS CloudHSM 管理ユーティリティ (CMU) の listUsers コマンドを使用して、 AWS CloudHSM クラスター内のすべてのユーザーを一覧表示します。listUsers を実行する際、ログインは不要であり、すべてのユーザータイプでユーザーをリストアップできます。

クラスター上のすべてのユーザーを一覧表示

1. 設定ツールで CMU 設定を更新します。

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
   ```

2. CMU を開始します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:
   \ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. listUsers を使用して、クラスター上のすべてのユーザーを一覧表示します。

   ```
   aws-cloudhsm > listUsers
   ```

   CMU は、クラスター上のすべてのユーザーを一覧表示します。

   ```
   Users on server 0(10.0.2.9):
   Number of users found:4
   ```

```
    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              AU              app_user                                NO
               0               NO
         2              CO              example_officer                         NO
               0               NO
         3              CU              example_user                            NO
               0               NO
Users on server 1(10.0.3.11):
Number of users found:4

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              AU              app_user                                NO
               0               NO
         2              CO              example_officer                         NO
               0               NO
         3              CU              example_user                            NO
               0               NO
Users on server 2(10.0.1.12):
Number of users found:4

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              AU              app_user                                NO
               0               NO
         2              CO              example_officer                         NO
               0               NO
         3              CU              example_user                            NO
               0               NO
```

listUsers の詳細情報は、listUsers を参照してください。

AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーパスワードを変更する

AWS CloudHSM 管理ユーティリティ (CMU) changePswdで を使用して、ハードウェアセキュリティモジュール (HSM) ユーザーのパスワードを変更します。

ユーザータイプとパスワードは大文字と小文字が区別されますが、ユーザー名では区別されません。

Crypto User (CU) と Appliance User (AU) は、自分のパスワードのみ変更できます。別のユーザーのパスワードを変更する場合、CO としてログインする必要があります。ただし、現在ログインしているユーザーのパスワードを変更することはできません。

自分のパスワードの変更

1. 設定ツールで CMU 設定を更新します。

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
   ```

2. CMU を開始します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:
   \ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. HSM へログインします。

   ```
   aws-cloudhsm > loginHSM CO admin co12345
   ```

   接続 CMU リストの数が、クラスター内の HSM 数と一致していることを確認します。一致しない場合、ログアウトして最初からやり直してください。

4. changePswd を使用してユーザー自身のパスワードを変更します。

   ```
   aws-cloudhsm > changePswd CO example_officer <new password>
   ```

   CMU は、パスワードの変更オペレーションについてプロンプトを表示します。

```
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?
```

5. 「**y**」と入力します。

   CMU は、パスワードの変更オペレーションについてプロンプトを表示します。

```
Changing password for example_officer(CO) on 3 nodes
```

別のユーザーのパスワードの変更

1. 設定ツールで CMU 設定を更新します。

   Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

   Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. CMU を開始します。

   Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

   Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. CO ユーザーとして HSM にログインします。

```
aws-cloudhsm > loginHSM CO admin co12345
```

接続 CMU リストの数が、クラスター内の HSM 数と一致していることを確認します。一致しない場合、ログアウトして最初からやり直してください。

4. changePswd を使用して別のユーザーのパスワードを変更します。

```
aws-cloudhsm > changePswd CU example_user <new password>
```

CMU は、パスワードの変更オペレーションについてプロンプトを表示します。

```
**********************CAUTION*****************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**********************************************************

Do you want to continue(y/n)?
```

5. 「**y**」と入力します。

CMU は、パスワードの変更オペレーションについてプロンプトを表示します。

```
Changing password for example_user(CU) on 3 nodes
```

changePswd の詳細情報は、[パスワードの変更] を参照してください。

AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーを削除する

AWS CloudHSM 管理ユーティリティ (CMU) deleteUserで を使用して、ハードウェアセキュリティモジュール (HSM) ユーザーを削除します。別のユーザーを削除する場合、CO としてログインする必要があります。

> **Tip**
>
> キーを所有している Crypto User (CU) を削除することはできません。

ユーザーの削除

1. 設定ツールで CMU 設定を更新します。

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
   ```

2. CMU を開始します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:
   \ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. CO ユーザーとして HSM にログインします。

   ```
   aws-cloudhsm > loginHSM CO admin co12345
   ```

   接続 CMU リストの数が、クラスター内の HSM 数と一致していることを確認します。一致しない場合、ログアウトして最初からやり直してください。

4. deleteUser を使用してユーザーを削除します。

   ```
   aws-cloudhsm > deleteUser CO example_officer
   ```

   CMU がユーザーを削除します。

   ```
   Deleting user example_officer(CO) on 3 nodes
   deleteUser success on server 0(10.0.2.9)
   deleteUser success on server 1(10.0.3.11)
   deleteUser success on server 2(10.0.1.12)
   ```

deleteUser の詳細情報は、deleteUser を参照してください。

管理ユーティリティを使用してユーザーの 2FA AWS CloudHSM を管理する

セキュリティを向上する場合、2 要素認証 (2FA) を設定して AWS CloudHSM クラスターを保護できます。Crypto Officer (CO) に対してのみ 2FA を有効化できます。

2FA 有効のハードウェアサービスモジュール (HSM) アカウントでクラスターにログインする場合、cloudhsm_mgmt_util (CMU) にパスワード (最初の要素、ユーザーが知っているもの) を指定します。CMU はトークンを提供し、トークンに署名を要求するプロンプトを表示します。2 番目の要素 (自分の持っているもの) を提供する場合、すでに作成して HSM ユーザーに紐づけしたキーペアからプライベートキーを使用してトークンに署名します。クラスターにアクセスする場合、署名付きトークンを CMU に指定します。

> **Note**
>
> Crypto User (CU) またはアプリケーションに対して 2FA を有効化できません。2 要素認証 (2FA) は CO ユーザーのみ対象です。

トピック

- AWS CloudHSM 管理ユーティリティを使用した AWS CloudHSM クラスターのクォーラム認証と 2FA
- AWS CloudHSM 管理ユーティリティ AWS CloudHSM を使用するための 2FA キーペアの要件
- AWS CloudHSM 管理ユーティリティユーザーに対して 2FA が有効になっているユーザーを作成する
- AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーの 2FA を管理する
- AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーの 2FA を無効にする
- AWS CloudHSM 管理ユーティリティを使用した 2FA の設定リファレンス

AWS CloudHSM 管理ユーティリティを使用した AWS CloudHSM クラスターのクォーラム認証と 2FA

クラスターは、クォーラム認証と 2 要素認証 (2FA) に同じキーを使用します。これは、2FA が有効なユーザーが M of N アクセスコントロール (MofN) に登録されていることを意味します。同じ HSM

ユーザーに対して 2FA 認証とクォーラム認証を正常に実行する際、次の点を考慮する必要があります。

- 現在、ユーザーに対してクォーラム認証を使用している場合は、クォーラムのユーザーに対して作成したものと同じキーペアを使用し、ユーザーに対して 2FA を有効化する必要があります。
- クォーラム認証ユーザーではない非 2FA ユーザーに 2FA 要件を追加する場合、そのユーザーを 2FA 認証で MofN ユーザーとして登録します。
- 2FA 要件を削除するか、クォーラム認証ユーザーでもある 2FA ユーザーのパスワードを変更する場合、クォーラムのユーザーのMofN ユーザーとしての登録も削除されます。
- 2FA 要件を削除するか、クォーラム認証ユーザーでもある 2FA ユーザーのパスワードを変更する場合、それでもそのユーザーがクォーラム認証に加わる必要がある の場合、当該ユーザーを MofN ユーザーとして再登録する必要があります。

認証の詳細情報は、CMU を使用してクォーラム認証を管理する を参照してください。

AWS CloudHSM 管理ユーティリティ AWS CloudHSM を使用するための 2FA キーペアの要件

AWS CloudHSM ハードウェアセキュリティモジュール (HSM) ユーザーの 2 要素認証 (2FA) を有効にするには、次の要件を満たすキーを使用します。

新しいキーペアの作成や、以下の要件を満たす既存のキーを使用することもできます。

- キータイプ : 非対称
- キーの使用方法 : 署名と認証
- キースペック: RSA_2,048
- 署名アルゴリズムには、以下が含まれます。
  - `sha256WithRSAEncryption`

> **Note**
>
> クォーラム認証を使用している場合、またはクォーラム認証を使用する予定の場合は、the section called “クォーラム認証” を参照してください。

AWS CloudHSM 管理ユーティリティユーザーに対して 2FA が有効になっているユーザーを作成する

AWS CloudHSM 管理ユーティリティ CMU (CMU) とキーペアを使用して、2 要素認証 (2FA) を有効にした新しい暗号局 (CO) ユーザーを作成します。

2FA を有効化した状態での CO ユーザー作成

1. 1 つのターミナルで、以下のステップを実行します。

   a. HSM にアクセスし、CloudHSM 管理ユーティリティにログインします。

   ```
   /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   b. CO としてログインし、以下のコマンドを使用して 2FA で新しいユーザー MFA を作成します。

   ```
   aws-cloudhsm > createUser CO MFA <CO USER PASSWORD> -2fa /home/ec2-user/
   authdata
   ************************CAUTION*********************************
   This is a CRITICAL operation, should be done on all nodes in the
   cluster. AWS does NOT synchronize these changes automatically with the
   nodes on which this operation is not executed or failed, please
   ensure this operation is executed on all nodes in the cluster.
   ****************************************************************

   Do you want to continue(y/n)? y

   Creating User exampleuser3(CO) on 1 nodesAuthentication data written to: "/
   home/ec2-user/authdata"Generate Base64-encoded signatures for SHA256 digests in
    the authentication datafile.
   To generate the signatures, use the RSA private key, which is the second factor
    ofauthentication for this user. Paste the signatures and the corresponding
    public keyinto the authentication data file and provide
   the file path below.Leave this field blank to use the path initially
    provided.Enter filename:
   ```

   c. 上記のターミナルは、この状態のままにしておきます。Enter キーを押したり、ファイル名を入力したりしないでください。

2. 別のターミナルで、以下のステップを実行します。

   a. HSM にアクセスし、CloudHSM 管理ユーティリティにログインします。

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

b. 以下のコマンドを使用して、パブリックプライベートキーペアを生成します。

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt
 rsa_keygen_bits:2048
```

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

c. 以下のコマンドを実行して、authdata ファイルからダイジェストを抽出するための JSON クエリ機能をインストールします。

```
sudo yum install jq
```

d. ダイジェスト値を抽出するには、まず authdata ファイル内の以下のデータを検索します。

```
{
  "Version":"1.0",
  "PublicKey":"",
  "Data":[
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature": ""
    }
  ]
}
```

> **Note**
>
> 取得したダイジェストは base64 でエンコードされていますが、ダイジェストに署名するには、まずファイルをデコードしてから署名する必要があります。次のコマンドはダイジェストをデコードし、デコードされたコンテンツを 「digest1.bin」に保存します
>
> ```
> cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |
>  base64 -d > digest1.bin
> ```

e. パブリックキーの内容を変換し、次に示すように「\n」を追加し、スペースを削除します。

```
-----BEGIN PUBLIC KEY-----\n<PUBLIC KEY>\n-----END PUBLIC KEY-----
```

> **Important**
>
> 上記のコマンドは、BEGIN PUBLIC KEY----- の直後に「\n」を追加する方法、「\n」とパブリックキーの最初の文字の間のスペースを削除する方法、-----END PUBLIC KEY の前に「\n」を追加する方法、および「\n」とパブリックキーの末尾の間のスペースを削除する方法を示しています。

これはパブリックキーの PEM 形式で、認証データファイルで受け入れられます。

f. パブリックキー PEM 形式のコンテンツを authdata ファイルのパブリックキーセクションに貼り付けます。

```
vi authdata
```

```
{
  "Version":"1.0",
  "PublicKey":"-----BEGIN PUBLIC KEY-----\n<"PUBLIC KEY">\n-----END PUBLIC
 KEY-----",
  "Data":[
    {
      "HsmId":<"HSM ID">,
      "Digest":<"DIGEST">,
      "Signature": ""
    }
  ]
}
```

g. 次のコマンドを使用してトークン ファイルに署名します。

```
openssl pkeyutl -sign -in digest1.bin -inkey private_key.pem -pkeyopt
 digest:sha256 | base64
Output Expected:
<"THE SIGNATURE">
```

> **Note**
>
> 上記のコマンドで示したように、署名には openssl dgst の代わりに openssl pkeyutl を使用してください。

h. Authdata ファイルの「署名」フィールドに署名済みダイジェストを追加します。

```
vi authdata
```

```
{
    "Version": "1.0",
    "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
    "Data": [
        {
            "HsmId": <"HSM ID">,
            "Digest": <"DIGEST">,
            "Signature": <"Kkdl ... rkrvJ6Q==">
        },
        {
            "HsmId": <"HSM ID">,
            "Digest": <"DIGEST">,
            "Signature": <"K1hxy ... Q261Q==">
        }
    ]
}
```

3. 最初のターミナルに戻り、**Enter** を押します。

```
Generate Base64-encoded signatures for SHA256 digests in the authentication
 datafile. To generate the signatures, use the RSA private key,
which is the second factor ofauthentication for this user. Paste the signatures and
 the corresponding public keyinto the authentication data file and provide the file
 path below. Leave this field blank to use the path initially provided.
Enter filename: >>>>> Press Enter here

createUser success on server 0(10.0.1.11)
```

AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーの 2FA を管理する

AWS CloudHSM 管理ユーティリティ (CMU) changePswdで を使用して、ユーザーの 2 要素認証 (2FA) を変更します。2FA を有効化にするたびに、2FA ログイン用のパブリックキーを指定する必要があります。

changePswd は、次のいずれかのシナリオを実行します。

- 2FA ユーザーのパスワード変更
- 非 2FA ユーザーのパスワード変更
- 非 2FA ユーザーに 2FA の追加
- 2FA ユーザーから 2FA の削除
- 2FA ユーザーのキーのローテーション化

また、タスクを組み合わせることもできます。たとえば、ユーザーから 2FA を削除すると同時にパスワードの変更や、2FA キーをローテーション化してユーザーパスワードの変更を行うことができます。

2FA が有効な CO ユーザーのパスワードの変更、またはキーのローテーション化

1. CMU を使用し、2FA が有効な CO として HSM にログインします。
2. changePswd を使用して、2FA が有効な CO ユーザーからパスワードを変更するか、キーをローテーション化します。-2fa パラメータを使用して、システムが authdata ファイルに書き込むファイルシステム内の位置を含みます。このファイルには、クラスター内の各 HSM のダイジェストが含まれています。

   ```
   aws-cloudhsm > changePswd CO example-user <new-password> -2fa /path/to/authdata
   ```

   CMU は、プライベートキーを使用して、authdata ファイル内のダイジェストに署名を要求するプロンプトが表示され、署名はパブリックキー付きで返却されます。
3. プライベートキーを使用して、authdata ファイル内のダイジェストに署名し、署名とパブリックキーを JSON 形式の authdata ファイルに追加後、CMU に authdata ファイルの位置を追加します。詳細情報は、the section called “設定リファレンス” を参照してください。

> **Note**
>
> クラスターは、クォーラム認証と 2FA に同じキーを使用します。クォーラム認証を使用している場合、またはクォーラム認証を使用する予定の場合は、the section called “クォーラム認証” を参照してください。

AWS CloudHSM 管理ユーティリティを使用して HSM ユーザーの 2FA を無効にする

AWS CloudHSM 管理ユーティリティ (CMU) を使用して、 のハードウェアセキュリティモジュール HSM) ユーザーの 2 要素認証 (2FA) を無効にします AWS CloudHSM。

2FA が有効な CO ユーザーの 2FA の無効化

1. CMU を使用し、2FA が有効な CO として HSM にログインします。
2. changePswd を使用して、2FA が有効な CO ユーザーから 2FA を削除します。

```
aws-cloudhsm > changePswd CO example-user <new password>
```

CMU は、パスワードの変更オペレーションを要求するプロンプトを表示します。

> **Note**
>
> 2FA 要件を削除するか、クォーラム認証ユーザーでもある 2FA ユーザーのパスワードを変更する場合、クォーラムのユーザーのMofN ユーザーとしての登録も削除されます。クォーラムユーザーおよび 2FA の詳細情報は、[the section called “クォーラム認証”] を参照してください。

3. 「**y**」と入力します。

CMU は、パスワードの変更オペレーションを確定します。

AWS CloudHSM 管理ユーティリティを使用した 2FA の設定リファレンス

以下は、 AWS CloudHSM 管理ユーティリティ (CMU) が生成したリクエストとレスポンスの両方について、 `authdata` ファイル内の 2 要素認証 (2FA) プロパティの例です。

```
{
```

```
    "Version": "1.0",
    "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
    "Data": [
        {
            "HsmId": "hsm-lgavqitns2a",
            "Digest": "k5O1p3f6foQRVQH7S8Rrjcau6h3TYqsSdr16A54+qG8=",
            "Signature": "Kkdl ... rkrvJ6Q=="
        },
        {
            "HsmId": "hsm-lgavqitns2a",
            "Digest": "IyBcx4I5Vyx1jztwvXinCBQd9lDx8oQe7iRrWjBAi1w=",
            "Signature": "K1hxy ... Q261Q=="
        }
    ]
}
```

[データ]

最上位のノード。クラスター内の各 HSM の下位ノードが含まれています。すべての 2FA コマンドのリクエストとレスポンスに表示されます。

ダイジェスト

これは、認証の 2 番目の要素を提供するために署名が必要です。すべての 2FA コマンドのリクエストで生成された CMU。

Hsmid

ご利用のHSM の IDです。すべての 2FA コマンドのリクエストとレスポンスに表示されます。

パブリックキー

生成したキーペアのパブリックキーの部分は、PEM 形式の文字列として挿入されます。createUser と changePswd の回答欄にこれを入力します。

署名

Base 64 でエンコードされた署名付きダイジェスト。2FA コマンドの回答欄にこれを入力します。

バージョン

認証データ JSON 形式のファイルのバージョン。すべての 2FA コマンドのリクエストとレスポンスに表示されます。

CloudHSM 管理ユーティリティ (CMU) を使用したクォーラム認証の管理 (M of N アクセスコントロール)

AWS CloudHSM クラスターHSMs は、M of N アクセスコントロールとも呼ばれるクォーラム認証をサポートしています。クォーラム認証を使用すると、HSM の単一のユーザーは HSM でクォーラム管理されたオペレーションを行うことができません。代わりに、HSM ユーザーの最小数 (少なくとも 2 人) が、これらのオペレーションを協力して行う必要があります。クォーラム認証を使用すると、複数の HSM ユーザーからの承認を要求することで、さらに保護レイヤーを追加できます。

クォーラム認証は次のオペレーションを制御できます。

- Crypto officer (CO) による HSM ユーザーの管理 HSM ユーザーの作成と削除、および、別の HSM ユーザーのパスワードの変更。詳細については、「管理ユーティリティでクォーラム認証を有効にしたユーザー AWS CloudHSM 管理」を参照してください。

AWS CloudHSMでのクォーラム認証の使用に関する次の追加の情報に注意してください。

- HSM ユーザーは自分のクォーラムトークンに署名できます。つまり、リクエストするユーザーはクォーラム認証に必要な承認の 1 つを提供できます。
- クォーラム管理されたオペレーションに対して、最小数のクォーラム承認者を選択します。選択できる最小数は 2 で、選択できる最大数は 8 です。
- HSM はクォーラムトークンを最大 1024 保存できます。HSM にすでに 1024 トークンある場合、新しく作成しようとすると、HSM は期限切れのトークンの 1 つを消去します。デフォルトでは、トークンは作成後 10 分で有効期限が切れます。
- クラスターは、クォーラム認証と 2 要素認証 (2FA) に同じキーを使用します。クォーラム認証と 2FA の使用の詳細については、クォーラム認証と 2FA を参照してください。

次のトピックでは、 AWS CloudHSMでのクォーラム認証についてさらに詳細な情報を提供します。

トピック

- AWS CloudHSM 管理ユーティリティのクォーラム認証プロセス
- AWS CloudHSM Crypto Officer のクォーラム認証を設定する
- 管理ユーティリティでクォーラム認証を有効にしたユーザー AWS CloudHSM 管理
- Management AWS CloudHSM Utility を使用してクォーラム最小値を変更する

AWS CloudHSM 管理ユーティリティのクォーラム認証プロセス

以下のステップは、クォーラム認証のプロセスの概要を示しています。特定のステップとツールについては、管理ユーティリティでクォーラム認証を有効にしたユーザー AWS CloudHSM 管理 を参照してください。

1. 各 HSM ユーザーは署名のための非対称キーを作成します。これは HSM の外部で行い、キーを適切に保護します。
2. 各 HSM ユーザーは HSM にログインし、署名キーの公開部分 (パブリックキー) を HSM に登録します。
3. HSM ユーザーがクォーラム管理されたオペレーションを実行する場合、各ユーザーが HSM にログインし、クォーラムトークンを取得します。
4. HSM ユーザーは、クォーラムトークンを 1 人または複数の他の HSM ユーザーに付与し、承認を求めます。
5. 他の HSM ユーザーは、キーを使用してクォーラムトークンに暗号で署名することにより承認します。これは HSM の外部で行われます。
6. HSM ユーザーは、必要な数の承認を得ると、HSM にログインし、クォーラムトークンと承認 (署名) を HSM に渡します。
7. HSM では、それぞれの署名した人の登録されたパブリックキーを使用して署名を確認します。署名が有効である場合、HSM はトークンを承認します。
8. HSM ユーザーはクォーラム管理されたオペレーションを実行できます。

AWS CloudHSM Crypto Officer のクォーラム認証を設定する

以下のトピックでは、 AWS CloudHSM 暗号責任者 (COs) がクォーラム認証を使用できるように、ハードウェアセキュリティモジュール (HSM) を設定するために完了する必要がある手順について説明します。CO のクォーラム認証を最初に設定する場合に、これらのステップを 1 回だけ実行する必要があります。これらのステップが完了したら、管理ユーティリティでクォーラム認証を有効にしたユーザー AWS CloudHSM 管理 を参照してください。

トピック

- 前提条件
- ステップ 1. 署名のためのキーの作成と登録
- ステップ 2. HSM のクォーラム最小値を設定する

前提条件

この例の理解には、cloudhsm_mgmt_util (CMU)コマンドラインツール についての知識が必要です。この例では、 listUsers コマンドからの次の出力に示すように、 AWS CloudHSM クラスターには 2 つの HSMs があり、それぞれが同じ COsを持ちます。ユーザー作成の詳細については、HSM ユーザー を参照してください。

```
aws-cloudhsm > listUsers
Users on server 0(10.0.2.14):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                NO
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              officer1                             NO
           0               NO
         4              CO              officer2                             NO
           0               NO
         5              CO              officer3                             NO
           0               NO
         6              CO              officer4                             NO
           0               NO
         7              CO              officer5                             NO
           0               NO
Users on server 1(10.0.1.4):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                NO
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              officer1                             NO
           0               NO
         4              CO              officer2                             NO
           0               NO
         5              CO              officer3                             NO
           0               NO
```

```
        6              CO             officer4                       NO
          0               NO
        7              CO             officer5                       NO
          0               NO
```

ステップ 1. 署名のためのキーの作成と登録

クォーラム認証を使用する場合、各 CO が以下の すべて のステップをを実行する必要があります。

トピック

- RSA キーペアの作成
- 登録トークンの作成と署名
- HSM でパブリックキーを登録する

RSA キーペアの作成

様々なキーペアを作成、保護する方法があります。次の例では、OpenSSL 使用方法を説明しています。

Example— OpenSSL でプライベートキーを作成する

次の例は、OpenSSL を使用してパスフレーズで保護された 2048 ビットの RSA キーを作成する方法を示しています。この例を使用するには、*officer1.key* を、キーの保存先のファイル名に置き換えてください。

```
$ openssl genrsa -out <officer1.key> -aes256 2048
        Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
Enter pass phrase for officer1.key:
Verifying - Enter pass phrase for officer1.key:
```

次に、作成したプライベートキーを使用してパブリックキーを生成します。

Example— OpenSSL でパブリックキーを作成する

以下の例は、OpenSSL を使用して先ほど作成したプライベートキーからパブリックキーを作成する方法を示しています。

```
$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
```

```
Enter pass phrase for officer1.key:
writing RSA key
```

登録トークンの作成と署名

トークンを作成し、前のステップで生成したプライベートキーを使用して署名します。

Example— トークンを作成する

登録トークンは、最大 245 バイトのサイズを超えないランダムなデータを含むファイルのみです。プライベートキーを使用してトークンに署名し、プライベートキーへのアクセス権があることを示します。次のコマンドは、echo を使用して文字列をファイルにリダイレクトします。

```
$ echo <token to be signed> > officer1.token
```

トークンに署名し、署名ファイルに保存します。HSM で MofN ユーザーとして CO を登録する場合、署名付きトークン、署名なしトークン、およびパブリックキーが必要です。

Exampleトークンへ署名する

OpenSSL とプライベートキーを使用して登録トークンに署名し、署名ファイルを作成します。

```
$ openssl dgst -sha256 \
    -sign officer1.key \
    -out officer1.token.sig officer1.token
```

HSM でパブリックキーを登録する

キーを作成した後、CO はキーのパブリックパート (パブリックキー) を HSM に登録する必要があります。

HSM にパブリックキーの登録するには

1. 次のコマンドを使用して、cloudhsm_mgmt_util コマンドラインツールをスタートします。

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. loginHSM コマンドを使用して、CO ユーザーとして HSM にログインします。詳細については、「???」を参照してください。
3. registerQuorumPubKey コマンドを使用してパブリックキーを登録します。詳細については、次の例を参照するか、または help registerQuorumPubKey コマンドを使用してください。

Example- HSM のパブリックキーを登録する

以下の例では、cloudhsm_mgmt_util コマンドラインツールで registerQuorumPubKey コマンドを使用して、CO のパブリックキーを HSM に登録する方法を示しています。このコマンドを使用するには、CO が HSM にログインしている必要があります。以下の値を自分の値に置き換えてください。

```
aws-cloudhsm > registerQuorumPubKey
 CO <officer1> <officer1.token> <officer1.token.sig> <officer1.pub>
*************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
***************************************************************

Do you want to continue(y/n)? y
registerQuorumPubKey success on server 0(10.0.2.14)
```

<officer1.token>

署名なし登録トークンを含むファイルへのパスです。最大ファイルサイズが 245 バイトの任意のランダムデータを持つことができます。

必須 : はい

<officer1.token.sig>

登録トークンの SHA256_PKCS メカニズム署名付きハッシュを含むファイルへのパスです。

必須: はい

<officer1.pub>

非対称 RSA-2048 キーペアの公開キーを含むファイルへのパスです。プライベートキーを使用して、登録トークンに署名します。

必須: はい

次の例に示すように、すべての CO がパブリックキーを登録した後、listUsers コマンドの出力の MofnPubKey 列にこれが表示されます。

```
aws-cloudhsm > listUsers
Users on server 0(10.0.2.14):
```

```
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                             NO
           0               NO
         2              AU              app_user                          NO
           0               NO
         3              CO              officer1                        YES
           0               NO
         4              CO              officer2                        YES
           0               NO
         5              CO              officer3                        YES
           0               NO
         6              CO              officer4                        YES
           0               NO
         7              CO              officer5                        YES
           0               NO
Users on server 1(10.0.1.4):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                             NO
           0               NO
         2              AU              app_user                          NO
           0               NO
         3              CO              officer1                        YES
           0               NO
         4              CO              officer2                        YES
           0               NO
         5              CO              officer3                        YES
           0               NO
         6              CO              officer4                        YES
           0               NO
         7              CO              officer5                        YES
           0               NO
```

ステップ 2. HSM のクォーラム最小値を設定する

CO のクォーラム認証を使用するには、CO が HSM に ログインして、m 値とも呼ばれるクォーラム最小値を設定する必要があります。これは、HSM ユーザー管理オペレーションを実行するために必要な CO 承認の最小数です。HSM 上の任意の CO は、署名用のキーを登録していない CO を含む

クォーラム最小値を設定できます。クォーラム最小値はいつでも変更できます。詳細情報は、 最小値を変更 を参照してください。

HSM のクォーラム最小値の設定

1. 次のコマンドを使用して、cloudhsm_mgmt_util コマンドラインツールをスタートします。

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. loginHSM コマンドを使用して、CO ユーザーとして HSM にログインします。詳細については、「???」を参照してください。
3. クォーラム最小値を設定する場合、setMValue コマンドを使用します。詳細については、次の例を参照するか、または help setMValue コマンドを使用してください。

Example HSM のクォーラム最小値を設定する

この例では、クォーラム最小値 2 を使用します。最大は HSM 上の CO の合計数で、2 から 8 までの任意の値を選択できます。この例では HSM に 6 つの CO がいるため、指定可能な最大値は 6 です。

次のコマンド例を使用するには、最後の数値 (*2*) を所望のフォーラム最小値に置き換えてください。

```
aws-cloudhsm > setMValue 3 <2>
*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)? y
Setting M Value(2) for 3 on 2 nodes
```

上記の例では、最初の数字 (3) は、クォーラム最小値を設定しようとしている HSM サービスを示しています。

次の表に、HSM サービス識別子とその名前、説明、およびサービスに含まれるコマンドを示しています。

サービス識別子	サービス名	サービスの説明	HSM コマンド
3	USER_MGMT	HSM ユーザー管理	• createUser • deleteUser • changePswd (別の HSM ユーザーのパスワードを変更した場合のみ適用されます)
4	MISC_CO	その他の CO サービス	• setMValue

サービスのクォーラム最小値を取得するには、次の例のように、getMValue コマンドを使用します。

```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

前述の getMValue コマンドの出力は、HSM ユーザー管理オペレーション (サービス 3) のクォーラム最小値が 2 になったことを示しています。

これらのステップが完了したら、管理ユーティリティでクォーラム認証を有効にしたユーザー AWS CloudHSM 管理 を参照してください。

管理ユーティリティでクォーラム認証を有効にしたユーザー AWS CloudHSM 管理

ハードウェアセキュリティモジュール AWS CloudHSM (HSM) の Ancrypto Officer (CO) は、HSM で次のオペレーションのクォーラム認証を設定できます。

- HSM ユーザーの作成
- HSM ユーザーの削除
- 別の HSM ユーザーのパスワードの変更

HSM をクォーラム認証用に設定した後では、CO が単独で HSM ユーザー管理オペレーションを実行することはできません。次の例は、CO が HSM で新しいユーザーを作成しようとしたときの出力

を示しています。コマンドは RET_MXN_AUTH_FAILED エラーとなり、クォーラム認証が失敗したことを示しています。

```
aws-cloudhsm > createUser CU user1 password
**************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)? y
Creating User user1(CU) on 2 nodes
createUser failed: RET_MXN_AUTH_FAILED
creating user on server 0(10.0.2.14) failed

Retry/Ignore/Abort?(R/I/A): A
```

HSM ユーザー管理オペレーションを実行するには、CO は以下のタスクを完了する必要があります。

1. クォーラムトークン の取得
2. 他の CO からの承認 (署名) の取得
3. HSM でのトークンの承認
4. HSM ユーザー管理オペレーションの実行

HSM をクォーラム認証用にまだ設定していない場合は、今すぐに設定してください。詳細については、「初回の設定」を参照してください。

ステップ 1. クォーラムトークンの取得

まず、CO は cloudhsm_mgmt_util コマンドラインツールを使用して クォーラムトークン をリクエストする必要があります。

クォーラムトークンを取得するには

1. 次のコマンドを使用して、cloudhsm_mgmt_util コマンドラインツールをスタートします。

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. loginHSM コマンドを使用して、CO ユーザーとして HSM にログインします。詳細については、「???」を参照してください。
3. getToken コマンドを使用してクォーラムトークンを取得します。詳細については、次の例を参照するか、または help getToken コマンドを使用してください。

Exampleクォーラムトークンを取得する

この例では、ユーザー名を officer1 とする CO のクォーラムトークンを取得し、そのトークンを `officer1.token` というファイルに保存します。この例のコマンドを使用するには、以下の値を独自のものに置き換えてください。

- *`officer1`* トークンを取得する CO の名前。HSM にログインし、このコマンドを実行している CO と同じであることが必要です。
- *`officer1.token`* クォーラムトークンを保存するファイルの名前。

次のコマンドで、3 は取得するトークンを使用できる サービス を識別します。この例のトークンは、HSM ユーザー管理オペレーション (サービス 3) で使用できます。詳細については、「ステップ 2. HSM のクォーラム最小値を設定する」を参照してください。

```
aws-cloudhsm > getToken 3 officer1 officer1.token
getToken success on server 0(10.0.2.14)
Token:
Id:1
Service:3
Node:1
Key Handle:0
User:officer1
getToken success on server 1(10.0.1.4)
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
```

ステップ 2. 承認 CO からの署名の取得

クォーラムトークンを持つ CO は、そのトークンを他の CO に承認してもらう必要があります。他の CO は、承認を与えるために、署名キーを使用してトークンを暗号で署名します。この署名は HSM 外で行われます。

トークンの署名にはさまざまな方法が使用されます。次の例では、OpenSSL を使用しています。別の署名ツールを使用する場合は、そのツールで必ず CO のプライベートキー (署名キー) を使用してトークンの SHA-256 ダイジェストに署名します。

Example承認 CO からの署名を取得する

この例では、トークン (officer1) を持つ CO に少なくとも 2 つの承認が必要です。以下のコマンド例では、2 つの CO が OpenSSL を使用してトークンに暗号で署名する方法を示します。

最初のコマンドでは、officer1 が自分のトークンに署名します。以下のコマンド例を使用するには、以下の値を独自のものに置き換えてください。

- *officer1.key* および *officer2.key* CO の署名キーが含まれているファイルの名前。
- *officer1.token.sig1* および *officer1.token.sig2* 署名を保存するファイルの名前。署名ごとに別のファイルに保存します。
- *officer1.token* CO が署名するトークンが格納されるファイルの名前。

```
$ openssl dgst -sha256 -sign officer1.key -out officer1.token.sig1 officer1.token
Enter pass phrase for officer1.key:
```

次のコマンドでは、officer2 が同じトークンに署名します。

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

ステップ 3. HSM での署名済みトークンの承認

CO は、他の CO から最小限の数の承認 (署名) を取得したら、署名済みトークンを HSM で承認する必要があります。

HSM で署名済みトークンの承認

1. トークン承認ファイルを作成します。詳細については、次の例を参照してください。
2. 次のコマンドを使用して、cloudhsm_mgmt_util コマンドラインツールをスタートします。

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

3. loginHSM コマンドを使用して、CO ユーザーとして HSM にログインします。詳細については、「???」を参照してください。
4. approveToken コマンドを使用して署名済みトークンを承認し、トークン承認ファイルを渡します。詳細については、次の例を参照してください。

Exampleトークン承認ファイルを作成し、署名済みトークンを HSM で承認する

トークン承認ファイルは、HSM で必要とされる特別な形式のテキストファイルです。このファイルには、トークン、その承認者、および承認者の署名が含まれます。トークン承認ファイルの例は次のとおりです。

```
# For "Multi Token File Path", type the path to the file that contains
# the token. You can type the same value for "Token File Path", but
# that's not required. The "Token File Path" line is required in any
# case, regardless of whether you type a value.
Multi Token File Path = officer1.token;
Token File Path = ;

# Total number of approvals
Number of Approvals = 2;

# Approver 1
# Type the approver's type, name, and the path to the file that
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer1;
Approval File = officer1.token.sig1;

# Approver 2
# Type the approver's type, name, and the path to the file that
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer2;
Approval File = officer1.token.sig2;
```

トークン承認ファイルの作成後、CO は cloudhsm_mgmt_util コマンドラインツールを使用して HSM にログインします。次に CO は approveToken コマンドを使用し、以下の例に示すように、トークンを承認します。*approval.txt* は、トークン承認ファイルの名前に置き換えてください。

```
aws-cloudhsm > approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)
```

このコマンドが成功すると、HSM でのクォーラムトークンの承認が完了します。トークンのステータスを確認するには、次の例に示すように、listTokens コマンドを使用します。コマンドの出力は、トークンに必要な数の承認があることを示しています。

トークンの有効期間は、トークンが HSM に保持される保証期間を示します。トークンの有効期間が過ぎた (ゼロ秒) 後でも、トークンを使用できます。

```
aws-cloudhsm > listTokens
=====================
    Server 0(10.0.2.14)
=====================
-------- Token - 0 ----------
Token:
Id:1
Service:3
Node:1
Key Handle:0
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

=====================
    Server 1(10.0.1.4)
=====================
-------- Token - 0 ----------
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
```

```
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

listTokens success
```

ステップ 4. ユーザー管理オペレーションでのトークンの使用

前のセクションで示したように、トークンに必要な数の承認を取得すると、CO は以下のいずれかの HSM ユーザー管理オペレーションを実行できます。

- createUser コマンドを使用して HSM ユーザーを作成する
- deleteUser コマンドを使用して HSM ユーザーを削除する
- changePswd コマンドを使用して別の HSM ユーザーのパスワードを変更する

これらのコマンドの詳しい使用方法については、HSM ユーザー を参照してください。

CO は、トークンを 1 つのオペレーションにのみ使用できます。そのオペレーションが成功すると、トークンは無効になります。別の HSM ユーザー管理オペレーションを行うには、新しいクォーラムトークンを取得し、承認者から新しい署名を取得して、その新しいトークンを HSM で承認する必要があります。

> **Note**
>
> MofN トークンは、現在のログインセッションが開いている間だけ有効です。cloudhsm_mgmt_util からログアウトするか、ネットワーク接続が切断された場合、トークンは無効になります。同様に、承認されたトークンは cloudhsm_mgmt_util 内でのみ使用でき、他のアプリケーションでの認証には使用できません。

次のコマンド例で、CO は HSM で新しいユーザーを作成しています。

```
aws-cloudhsm > createUser CU user1 <password>
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************
```

```
Do you want to continue(y/n)? y
Creating User user1(CU) on 2 nodes
```

前のコマンドが成功すると、後続の listUsers コマンドに新しいユーザーが表示されます。

```
aws-cloudhsm > listUsers
Users on server 0(10.0.2.14):
Number of users found:8

    User Id             User Type       User Name                      MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                           NO
           0               NO
         2              AU              app_user                        NO
           0               NO
         3              CO              officer1                       YES
           0               NO
         4              CO              officer2                       YES
           0               NO
         5              CO              officer3                       YES
           0               NO
         6              CO              officer4                       YES
           0               NO
         7              CO              officer5                       YES
           0               NO
         8              CU              user1                           NO
           0               NO
Users on server 1(10.0.1.4):
Number of users found:8

    User Id             User Type       User Name                      MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                           NO
           0               NO
         2              AU              app_user                        NO
           0               NO
         3              CO              officer1                       YES
           0               NO
         4              CO              officer2                       YES
           0               NO
         5              CO              officer3                       YES
           0               NO
```

```
        6              CO             officer4                       YES
          0               NO
        7              CO             officer5                       YES
          0               NO
        8              CU             user1                           NO
          0               NO
```

CO が別の HSM ユーザー管理オペレーションを実行しようとすると、次の例に示すように、クォーラム認証エラーが発生して失敗します。

```
aws-cloudhsm > deleteUser CU user1
Deleting user user1(CU) on 2 nodes
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 0(10.0.2.14)

Retry/rollBack/Ignore?(R/B/I): I
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 1(10.0.1.4)

Retry/rollBack/Ignore?(R/B/I): I
```

listTokens コマンドは、つぎの例のように、CO に承認済みトークンがないことを示しています。別の HSM ユーザー管理オペレーションを実行するには、新しいクォーラムトークンを取得し、承認者から新しい署名を取得して、その新しいトークンを HSM で承認する必要があります。

```
aws-cloudhsm > listTokens
=====================
    Server 0(10.0.2.14)
=====================
Num of tokens = 0

=====================
    Server 1(10.0.1.4)
=====================
Num of tokens = 0

listTokens success
```

Management AWS CloudHSM Utility を使用してクォーラム最小値を変更する

AWS CloudHSM 暗号責任者 (COs) がクォーラム認証を使用できるようにクォーラム最小値を設定したら、クォーラム最小値を変更することができます。承認者の数が現在のクォーラム最小値以上の場

合のみ、HSM はクォーラム最小値の変更を許可します。たとえば、クォーラム最小値が 2 の場合、クォーラム最小値を変更するには最低 2 つの CO の承認が必要です。

クォーラム最小値の変更のためにクォーラムの承認を取得する場合、setMValue コマンド (サービス 4) の クォーラムトークン が必要です。setMValue コマンド (サービス 4) のクォーラムトークンを取得するには、サービス 4 のクォーラム最小値を 1 より大きくする必要があります。つまり、CO (サービス 3) のクォーラム最小値を変更するには、サービス 4 のクォーラム最小値の変更が必要になる場合があります。

次の表に、HSM サービス識別子とその名前、説明、およびサービスに含まれるコマンドを示しています。

サービス識別子	サービス名	サービスの説明	HSM コマンド
3	USER_MGMT	HSM ユーザー管理	• createUser • deleteUser • changePswd (別の HSM ユーザーのパスワードを変更した場合のみ適用されます)
4	MISC_CO	その他の CO サービス	• setMValue

Crypto Officer のクォーラム最小値を変更するには

1. 次のコマンドを使用して、cloudhsm_mgmt_util コマンドラインツールをスタートします。

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. loginHSM コマンドを使用して、CO ユーザーとして HSM にログインします。詳細については、「???」を参照してください。
3. getMValue コマンドを使用して、サービス 3 のクォーラム最小値を取得します。詳細については、次の例を参照してください。
4. getMValue コマンドを使用して、サービス 4 のクォーラム最小値を取得します。詳細については、次の例を参照してください。

5. サービス 4 のクォーラム最小値がサービス 3 の値より小さい場合、setMValue コマンドを使用してサービス 4 の値を変更します。サービス 4 の値を、サービス 3 の値と同じまたはそれより大きい値に変更します。詳細については、次の例を参照してください。
6. サービス 4 をトークンを使用できるサービスとして指定するように注意し、クォーラムトークン を取得 します。
7. 他の CO からの承認 (署名) の取得
8. HSM でのトークンの承認
9. setMValue コマンドを使用して、サービス 3 (CO が実行するユーザー管理オペレーション) のクォーラム最小値を変更します。

Example- クォーラム最小値を取得してサービス 4 の値を変更する

次のコマンド例では、サービス 3 のクォーラム最小値が現在 2 であることを示しています。

```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

次のコマンド例では、サービス 4 のクォーラム最小値が現在 1 であることを示しています。

```
aws-cloudhsm > getMValue 4
MValue of service 4[MISC_CO] on server 0 : [1]
MValue of service 4[MISC_CO] on server 1 : [1]
```

サービス 4 のクォーラム最小値を変更するには、サービス 3 の値と同じがそれより大きい値を設定して setMValue コマンドを使用します。次の例では、サービス 4 のクォーラム最小値をサービス 3 に設定されているのと同じ値である 2 に設定します。

```
aws-cloudhsm > setMValue 4 2
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)? y
Setting M Value(2) for 4 on 2 nodes
```

次のコマンド例は、サービス 3 とサービス 4 でクォーラム最小値が現在 2 であることを示しています。

```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

```
aws-cloudhsm > getMValue 4
MValue of service 4[MISC_CO] on server 0 : [2]
MValue of service 4[MISC_CO] on server 1 : [2]
```

のキー AWS CloudHSM

AWS CloudHSM クラスターを暗号化処理に使用する前に、クラスターのハードウェアセキュリティモジュール (HSM) でユーザーとキーを作成する必要があります。

で AWS CloudHSM、次のいずれかを使用して、クラスター内の HSMsのキーを管理します。

- PKCS #11 ライブラリ
- JCE プロバイダー
- CNG および KSP プロバイダー
- CloudHSM CLI

キーを管理する前には、Crypto User (CU) のユーザー名とパスワードを使用して HSM にログインします。CU だけがキーを作成できます。キーを作成した CU は、その鍵を所有および管理します。

AWS CloudHSMでのキーの管理の詳細ついては、次のトピックを参照してください。

トピック

- でのキー同期と耐久性の設定 AWS CloudHSM
- での AES キーラッピング AWS CloudHSM
- での信頼されたキーの使用 AWS CloudHSM
- CloudHSM CLI によるキー管理
- KMU AWS CloudHSM によるキー管理

でのキー同期と耐久性の設定 AWS CloudHSM

AWS CloudHSM は、作成したすべてのトークンキーを同期します。キーの同期は主に自動プロセスですが、クラスター内で最低 2 つの ハードウェアセキュリティモジュール (HSM) を使用して、キーの耐久性を高めることができます。このトピックでは、キーの同期設定、お客様がクラスター上でキーの操作を用いて直面する一般的な問題、およびキーの耐久性を高めるための戦略について説明します。

このトピックでは AWS CloudHSM、 でのキー同期設定、お客様がクラスターでキーを操作する際に直面する一般的な問題、およびキーをより耐久性の高いものにするための戦略について説明します。

トピック

- AWS CloudHSM 主要な概念
- AWS CloudHSM キー同期について
- AWS CloudHSM クライアントキーの耐久性設定を変更する
- クローンされた AWS CloudHSM クラスター間でキーを同期する

AWS CloudHSM 主要な概念

以下は AWS CloudHSMでのキー操作に関する基礎知識です。

Token keys

キー中に作成する永続キーは、オペレーションを生成、インポート、またはラップ解除します。は、クラスター全体でトークンキーを AWS CloudHSM 同期します。

Session keys

クラスター内の 1 つのハードウェアセキュリティモジュール (HSM) にのみ存在する工フェメラルキー。 AWS CloudHSM は、クラスター間でセッションキーを同期しません。

Client-side key synchronization

キーの生成、インポート、またはアンラップ操作中に作成したトークンキーをクローンするクライアント側のプロセス。少なくとも 2 つの HSM を用いてクラスターを実行することで、トークンキーの耐久性を高めることができます。

Server-side key synchronization

クラスター内のすべての HSM に対して定期的にキーをクローンします。管理は必要ありません。

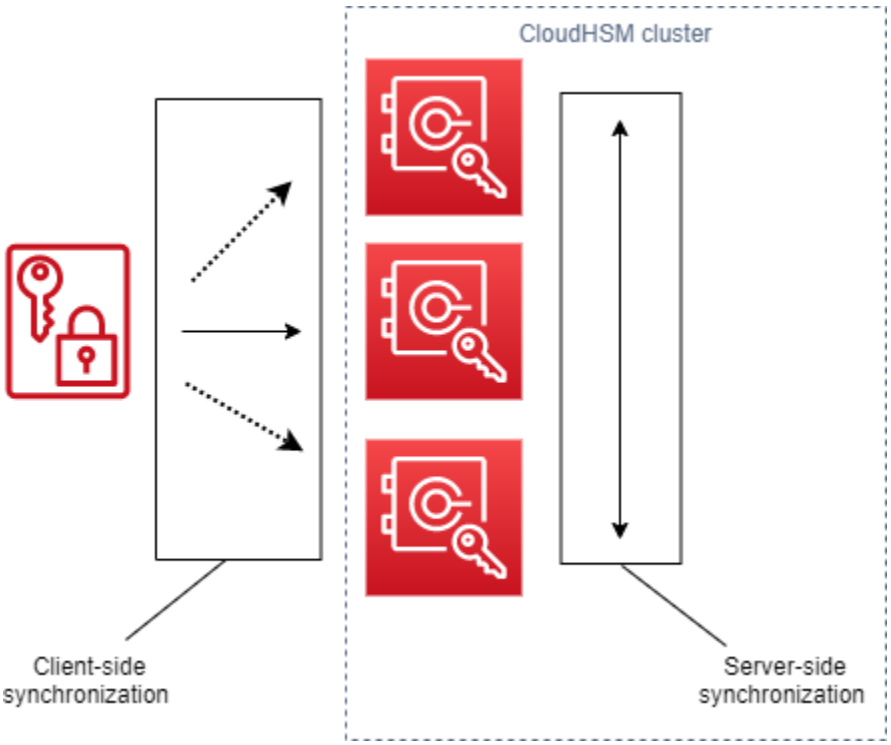
Client key durability settings

キーの耐久性に影響するクライアント上に構成した設定。[クライアント SDK 5] と [クライアント SDK 3] では動作が異なります。

- クライアント SDK 5 では、この設定を使用して、単一の HSM クラスターを実行します。
- クライアント SDK 3 では、この設定を使用して、キー作成オペレーションを成功させるために必要な HSM の数を指定します。

AWS CloudHSM キー同期について

AWS CloudHSM はキー同期を使用して、クラスター内のすべてのハードウェアセキュリティモジュール (HSM) でトークンキーのクローンを作成します。キーの生成、インポート、またはアンラップ操作中、トークンキーを永続キーとして作成します。クラスターを通してこれらのキーを配信するには、CloudHSM はクライアント側とサーバー側の両方にキーの同期を提供します。



キー同期の目標 (サーバー側とクライアント側の両方) は、新しいキーを作成した後、できるだけ迅速にクラスター全体に、それを配信することです。これは、新しいキーを使用するために実行する後続の呼び出しが、クラスター内の利用可能な HSM にルーティングされる可能性があるため重要です。発信したコールがキーなしで HSM にルーティングされた場合、コールは失敗します。キー作成操作の後に実行される後続の呼び出しをアプリケーションで再試行するように指定することで、これらのタイプの障害を軽減できます。同期に必要な時間は、クラスターやその他の無形オブジェクト上のワークロードによって異なります。CloudWatch メトリクスを使用して、このタイプの状況でアプリケーションが採用すべきタイミングを判断します。CloudWatch のメトリクスの詳細については、CloudWatch Metrics を参照してください。

クラウド環境でのキー同期の課題は、キーの耐久性です。1 つの HSM でキーを作成し、多くの場合、それらのキーを使ってすぐに開始します。キーがクラスター内の別のHSMに複製される前に、その上にキーを作成する HSM に障害が発生した場合は、キーで暗号化されたいかなるものに対するキー および アクセスを失います。このリスクを軽減するために、client-side synchronization を提供します。クライアント側の同期は、キーの生成、インポート、またはアンラップ操作中に作成した

キーをクローンするクライアント側のプロセスです。クローニングキーの作成時にクローンを作成すると、キーの耐久性が向上します。もちろん、1 つの HSM を用いてクラスター内のキーのクローンを作成することはできません。キーの耐久性を高めるために、最低 2 つの HSM を使用するようにクラスターを構成することをお勧めします。クライアント側の同期と 2 つの HSM を持つクラスターで、クラウド環境におけるキーの耐久性の課題に対応できます。

AWS CloudHSM クライアントキーの耐久性設定を変更する

キーの同期は主に自動プロセスですが、クライアント側のキーの耐久性設定を管理できます。クライアント側のキーの耐久性設定は、クライアント SDK 5 とクライアント SDK 3 では異なる動作をします。

- クライアント SDK 5 では、最低 2 つの HSM を用いてクラスターを実行することを求める key availability quorums のコンセプトを紹介します。クライアント側のキーの耐久性設定を使用して、2 つの HSM 要件をオプトアウトできます。quorums のさらなる詳細については、the section called “概念” を参照してください。
- クライアント SDK 3 では、クライアント側のキーの耐久性設定を使用して、オペレーション全体を成功と見なすためにキーの作成を成功させる必要がある HSM の数を指定します。

クライアント SDK 5 のクライアントキーの耐久性設定

クライアント SDK 5 では、キーの同期は完全自動プロセスです。キー可用性クォーラムを用いて、アプリケーションがキーを使用できるようになる前に、新しく作成されたキーがクラスター内の 2 つの HSM 上に存在している必要があります。キーの可用性クォーラムを使用するには、クラスターに最低 2 つの HSM が必要です。

クラスター構成がキーの耐久性要件を満たしていない場合、トークンキーを作成または使用しようとすると、ログに次のエラーメッセージが表示されて失敗します。

```
Key <key handle> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

クライアント構成設定を使用して、キーの可用性クォーラムをオプトアウトできます。たとえば、単一の HSM を用いてクラスターの実行をオプトアウトしたい場合があります。

クライアント SDK 5 の概念

Key Availability Quorum

AWS CloudHSM は、アプリケーションがキーを使用する前にキーが存在する必要があるクラスター内の HSMs の数を指定します。最低 2 つの HSM を持つクラスターが必要です。

クライアントキーの耐久性設定の管理

クライアントキーの耐久性設定を管理するには、クライアント SDK 5 用の構成ツールを使用する必要があります。

PKCS #11 library

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-
key-availability-check
```

OpenSSL Dynamic Engine

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

OpenSSL Dynamic Engine Provider

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --disable-key-availability-check
```

Key Storage Provider (KSP)

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-key-availability-check
```

JCE provider

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-availability-check
```

CloudHSM CLI

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-
key-availability-check
```

クライアント SDK 3 のクライアントキーの耐久性設定

[クライアント SDK 3] では、キーの同期は主に自動プロセスですが、クライアントキーの耐久性設定を使用してキーの耐久性を高めることができます。オペレーション全体を成功と見なすためにキーの作成を引き継ぐ必要がある HSM の数を指定します。クライアント側の同期では、選択した設定に関係なく、クラスター内のすべての HSM にキーのクローンするために、ベストエフォートの試行を行います。この設定では、指定した HSM の数の上にキーの作成が強制されます。値を指定し、システムがその数の HSM にキーをレプリケートできない場合、不要なキーマテリアルは自動的にクリーンアップされ、再試行できます。

> **Important**
>
> クライアントキーの耐久性設定を設定しない場合 (またはデフォルト値の 1 を使用する場合)、キーが失われる恐れがあります。サーバーサイドサービスがそのキーを別の HSM に複製する前に現在の HSM に障害が発生した場合、キーマテリアルは失われます。

キーの耐久性を最大化するには、クライアント側の同期のために少なくとも 2 つの HSM を指定することを検討してください。HSM をいくつ指定しても、クラスター上のワークロードは変わりません。クライアント側の同期では、クラスター内のすべての HSM に対して常にベストエフォート型のキーのクローンが試行されます。

推奨事項

- Minimum: クラスターあたり 2 つの HSM
- Maximum: クラスター内の HSM の総数より 1 少ない

クライアント側の同期に失敗すると、クライアントサービスは作成された可能性があり、不要になった可能性のある不要なキーをクリーンアップします。このクリーンアップは、常に機能するとは限らないベストエフォート対応です。クリーンアップが失敗した場合は、不要なキーマテリアルを削除する必要があります。さらなる詳細については、Key Synchronization Failures を参照してください。

クライアントキーの耐久性のための設定ファイルの設定

クライアントキーの耐久性設定を指定するには、cloudhsm_client.cfg を編集します。

クライアント設定ファイルを編集するには

1. cloudhsm_client.cfg を開きます。

   Linux ：

   ```
   /opt/cloudhsm/etc/cloudhsm_client.cfg
   ```

   Windows ：

   ```
   C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
   ```

2. ファイルのclientノードで、キー作成オペレーションを成功させるために がキーを正常に作成 AWS CloudHSM する必要がある HSMs の最小数の値を追加してcreate_object_minimum_nodes指定します。

   ```
   "create_object_minimum_nodes" : 2
   ```

   > **Note**
   >
   > [key_mgmt_util (KMU)]コマンドラインツール]には、クライアントキーの耐久性の追加設定があります。詳細については、the section called “KMU とクライアント側の同期” を参照してください。

設定リファレンス

これらは、cloudhsm_client.cfg の抜粋中に示されるクライアント側の同期プロパティです:

```
{
    "client": {
        "create_object_minimum_nodes" : 2,
        ...
    },

    ...
}
```

create_object_minimum_nodes

キーの生成、キーのインポート、またはキーアンラップ操作が成功したと見なすために必要な HSM の最小数を指定します。設定されると、デフォルトは、[1] になります。つまり、キーがオペレーション作成するたびに、クライアント側のサービスはクラスター内のすべての HSM でキーを作成しようとしますが、成功を返すためには、クラスター内の HSM 上で single key の生成を必要とするだけです。

KMU とクライアント側の同期

[key_mgmt_util (KMU)] コマンドラインツールを使用してキーを作成する場合は、オプションのコマンドラインパラメータ (-min_srv) を使って、キーのクローンを作成する HSM の数を limit します。コマンドラインパラメータと値を設定ファイルで指定すると、 は 2 つの値の LARGER を AWS CloudHSM 優先します。

詳細については、以下の各トピックを参照してください。

- genDSAKeyPair
- genECCKeyPair
- genRSAKeyPair
- genSymKey
- importPrivateKey
- importPubKey
- imSymKey
- insertMaskedObject

- unWrapKey

クローンされた AWS CloudHSM クラスター間でキーを同期する

クライアント側とサーバー側の同期は、同じ AWS CloudHSM クラスター内のキーの同期のみを目的としています。クラスターのバックアップを別のリージョンにコピーする場合は、キーレプリケートコマンドを使用して 2 つのクラスター間でキーをレプリケートします。クローンクラスターは、クロスリージョンの冗長性のため、または災害対策プロセスを簡素化するために使用できます。CloudHSM CLI をインストールしていない場合は、??? の手順を参照してください。

での AES キーラッピング AWS CloudHSM

このトピックでは、AES キーラッピングのオプションについて説明します AWS CloudHSM。AES キーラップは、AES キー (ラップキー) を使用して、任意のタイプの別のキー (ターゲットキー) をラップします。キーラップを使用して、保存されているキーを保護したり、安全でないネットワーク上でキーを送信したりします。

トピック

- サポートされているアルゴリズム
- での AES キーラップの使用 AWS CloudHSM

サポートされているアルゴリズム

AWS CloudHSM には AES キーラッピング用の 3 つのオプションがあり、それぞれがラップする前にターゲットキーをパディングする方法に基づいています。パディングは、キーラップを呼び出すときに、使用するアルゴリズムに従って自動的に行われます。次の表は、サポートされているアルゴリズムと関連する詳細の一覧で、アプリケーションに適したラップメカニズムを選択するのに役立ちます。

AES キーラップアルゴリズム	の仕様	サポートされているターゲットキーのタイプ	パディングスキーム	AWS CloudHSM クライアントの可用性
ゼロパディングを使用する AES キーラップ	RFC 5649 と SP 800–38F	すべて	必要に応じて、キービットの後にゼロを追加し	SDK 3.1 以降

AES キーラップアルゴリズム	の仕様	サポートされているターゲットキーのタイプ	パディングスキーム	AWS CloudHSM クライアントの可用性
			てブロック整列する	
パディングなしの AES キーラップ	RFC 3394 と SP 800–38F	AES や 3DES などのブロック整列キー	なし	SDK 3.1 以降
PKCS #5 パディングを使用する AES キーラップ	なし	すべて	ブロック整列するために、PKCS #5 パディングスキームに従って最低 8 バイトが追加されます	すべて

上記の表の AES キーラップアルゴリズムをアプリケーションで使用する方法については、「AWS CloudHSMでの AES キー ラップの使用」を参照してください。

AES キーラップの初期化ベクトルについて

ラップの前に、CloudHSM はデータの整合性を保つためにターゲットキーに初期化ベクトル (IV) を追加します。各キーラップアルゴリズムには、許可される IV のタイプに特定の制限があります。IV を設定するには AWS CloudHSM、次の 2 つのオプションがあります。

- 暗黙的: IV を NULL に設定すると、CloudHSM はラップおよびラップ解除オペレーションのためにそのアルゴリズムのデフォルト値を使用します (推奨)
- 明示的: デフォルトの IV 値をキーラップ関数に渡すことによって IV を設定します

> **Important**
>
> アプリケーションで使用している IV を理解する必要があります。キーをラップ解除するには、キーをラップするために使用したのと同じ IV を指定する必要があります。暗黙的な IV

を使用してラップする場合は、暗黙的な IV を使用してラップ解除します。暗黙的な IV では、CloudHSM はデフォルト値を使用してラップ解除します。

次の表に、ラップアルゴリズムで指定する IV の許容値を示します。

AES キーラップアルゴリズム	暗黙的な IV	明示的な IV
ゼロパディングを使用する AES キーラップ	必須 デフォルト値: (IV は仕様に基づいて内部で計算されます)	許可されていません
パディングなしの AES キーラップ	許可 (推奨) デフォルト値: `0xA6A6A6A6A6A6A6A6`	許可されています この値のみが受け入れられます: `0xA6A6A6A6A6A6A6A6`
PKCS #5 パディングを使用する AES キーラップ	許可 (推奨) デフォルト値: `0xA6A6A6A6A6A6A6A6`	許可されています この値のみが受け入れられます: `0xA6A6A6A6A6A6A6A6`

での AES キーラップの使用 AWS CloudHSM

次のようにキーをラップおよびラップ解除します。

- PKCS #11 library] 中で、次の表に示すように `C_WrapKey` および `C_UnWrapKey` の関数のために適切なメカニズムを選択します。
- JCE provider では、次の表に示すように、適切なアルゴリズム、モードとパディングの組み合わせ、暗号メソッド `Cipher.WRAP_MODE` および `Cipher.UNWRAP_MODE` の実装を選択します。
- CloudHSM CLI で、次の表に示すように、サポートされている CloudHSM CLI のキーラップコマンド と CloudHSM CLI のキーアンラップコマンド アルゴリズムのリストから適切なアルゴリズムを選択します。
- key_mgmt_util (KMU) では、次の表に示すように KMU を使用して AWS CloudHSM キーをエクスポートする および KMU を使用して AWS CloudHSM キーをラップ解除する コマンドを適切な m 値とともに使用します。

AES キーラップアルゴリズム	PKCS #11 メカニズム	Java メソッド	CloudHSM CLI サブコマンド	キー管理ユーティリティ (KMU) の引数
ゼロパディングを使用する AES キーラップ	• CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD (ベンダー定義のメカニズム)	AESWrap/ECB/ZeroPadding	aes-zero-pad	m = 6
パディングなしの AES キーラップ	• CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD (ベンダー定義のメカニズム)	AESWrap/ECB/NoPadding	aes-no-pad	m = 5
PKCS #5 パディングを使用する AES キーラップ	• CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD (ベンダー定義のメカニズム)	AESWrap/ECB/PKCS5Padding	aes-pkcs5-pad	m = 4

での信頼されたキーの使用 AWS CloudHSM

AWS CloudHSM は、内部者の脅威からデータキーを保護するために、信頼できるキーラッピングをサポートしています。このトピックでは、データを保護する信頼できるキーの作り方を説明します。

トピック

- での信頼されたキーについて AWS CloudHSM
- の信頼されたキー属性 AWS CloudHSM
- 信頼されたキーを使用して でデータキーをラップする方法 AWS CloudHSM
- の信頼されたキーを使用してデータキーをラップ解除する方法 AWS CloudHSM

での信頼されたキーについて AWS CloudHSM

信頼できるキーとは、他のキーをラップするために使用されるキーで、管理者や暗号化担当者 (CO) がその属性 CKA_TRUSTED を使用して信頼できるキーであることを明確に識別します。さらに、管理者と暗号責任者 (CO) は、CKA_UNWRAP_TEMPLATE と関連する属性を使用して、信頼できるキーによってデータキーがラップ解除された後に実行できるアクションを指定します。ラップ解除オペレーションを正常に行うには、信頼できるキーによってラップ解除されたデータキーにもこれらの属性が含まれている必要があります。これにより、ラップされていないデータキーは意図した用途でのみ許可されることが保証されます。

信頼できるキーを用いてラップするすべてのデータキーを識別するために、属性 CKA_WRAP_WITH_TRUSTED を使用します。こうすることで、アプリケーションがそれらをアンラップするのに信頼できるキーのみを使えるように、データキーを制限します。データキーにこの属性を設定すると、その属性は読み取り専用となり、変更することができなくなります。これらの属性を配置すると、アプリケーションは信頼するキーでのみデータキーをアンラップできます。そしてアンラップは、常にこれらのデータキーの使用を制限しようとする属性を持つデータキーをもたらします。

の信頼されたキー属性 AWS CloudHSM

次の属性を使用すると、 AWS CloudHSM キーを信頼済みとしてマークし、データキーを信頼済みキーでラップおよびラップ解除することのみを指定し、ラップ解除後にデータキーが実行できる操作を制御できます。

- CKA_TRUSTED: この属性 (CKA_UNWRAP_TEMPLATE に加えて) をデータキーをラップするキーに適用して、管理者または Crypto Officer (CO) が必要な努力を行っており、このキーを信頼していることを指定します。CKA_TRUSTED を設定できるのは管理者か CO だけです。Crypto User (CU) がそのキーを所有しますが、CKA_TRUSTED 属性を設定できるのは CO のみです。
- CKA_WRAP_WITH_TRUSTED: この属性を エクスポート可能なデータキーに適用して、このキーを CKA_TRUSTED としてマークされたキーでのみラップできるように指定します。いったん CKA_WRAP_WITH_TRUSTED を true に設定すると、属性は読み取り専用となり、属性を変更または削除することはできません。
- CKA_UNWRAP_TEMPLATE: この属性をラッピングキーに適用し (CKA_TRUSTED に加えて)、サービスがアンラップするデータキーにサービスを自動的に適用する必要がある属性名と値を指定します。アプリケーションがラップ解除用のキーを送信するとき、ラップ解除テンプレートを個別に指定できます。アンラップテンプレートを指定し、アプリケーションが独自のアンラップテンプレートを提供する場合、HSM は両方のテンプレートを使用して属性名と値をキーに適用します。ただ

し、ラッピングキーの CKA_UNWRAP_TEMPLATE の中の値が、アンラップ要求中にアプリケーションによって提供された属性と競合する場合、アンラップ要求は失敗します。

属性の詳細については、次のトピックを参照してください。

- PKCS #11 の主要属性
- JCE キーの属性
- CloudHSM CLI のキー属性

信頼されたキーを使用して でデータキーをラップする方法 AWS CloudHSM

信頼されたキーを使用してデータキーをラップするには AWS CloudHSM、3 つの基本的なステップを完了する必要があります。

1. 信頼できるキーでラップする予定のデータキーについては、その CKA_WRAP_WITH_TRUSTED 属性を true に設定します。
2. データキーをラップする予定の信頼できるキーについては、その CKA_TRUSTED 属性を true に設定します。
3. 信頼できるキーを使用してデータキーをラップします。

ステップ 1: データキーの **CKA_WRAP_WITH_TRUSTED** を true に設定する

ラップしたいデータキーについて、以下のオプションのいずれかを選択してキーの CKA_WRAP_WITH_TRUSTED 属性を true に設定します。こうすることで、アプリケーションがそれらをラップするのに信頼できるキーのみを使えるように、データキーを制限します。

オプション 1: 新しいキーを生成する場合は、**CKA_WRAP_WITH_TRUSTED** を true に設定する

PKCS #11、JCE、または CloudHSM CLI を使用してキーを生成します。詳細については、次の例を参照してください。

PKCS #11

PKCS #11 でキーを生成するには、キーの CKA_WRAP_WITH_TRUSTED 属性を true に設定する必要があります。次の例に示すように、これを行うには、この属性をキーの CK_ATTRIBUTE template に含めてから、属性を true に設定します。

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
        {CKA_WRAP_WITH_TRUSTED, &true_val,         sizeof(CK_BBOOL)},
        {CKA_LABEL,             label,             strlen(label)},
        ...
};
```

詳細については、PKCS #11 によるキーの生成を実演する公開サンプルを参照してください。

JCE

JCE でキーを生成するには、キーの WRAP_WITH_TRUSTED 属性を true に設定する必要があります。次の例に示すように、これを行うには、この属性をキーの KeyAttributesMap に含めてから、属性を true に設定します。

```
final String label = "test_key";
final KeyAttributesMap keySpec = new KeyAttributesMap();
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);
keySpec.put(KeyAttribute.LABEL, label);
...
```

詳細については、JCE によるキーの生成のデモを行う公開サンプルを参照してください。

CloudHSM CLI

CloudHSM CLI でキーを生成するには、キーの `wrap-with-trusted` 属性を true に設定する必要があります。これを行うには、キー生成コマンドの適切な引数に `wrap-with-trusted=true` を含めます。

- 対称キーの場合は、`attributes` 引数に `wrap-with-trusted` を追加します。
- パブリックキーの場合は、`public-attributes` 引数に `wrap-with-trusted` を追加します。
- プライベートキーの場合は、`private-attributes` 引数に `wrap-with-trusted` を追加します。

キーペアの生成の詳細については、「CloudHSM CLI の generate-asymmetric-pair カテゴリ」を参照してください。

対称キーの生成の詳細については、「CloudHSM CLI の generate-symmetric カテゴリ」を参照してください。

オプション 2: 既存のキーを使用する場合は、CloudHSM CLI を使用して **CKA_WRAP_WITH_TRUSTED** を true に設定する

既存のキーの CKA_WRAP_WITH_TRUSTED 属性を true に設定するには、以下の手順に従います。

1. CloudHSM CLI を使用して HSM にログインする コマンドを使用して、Crypto User (CU) としてログインします。
2. CloudHSM CLI でキーの属性を設定する コマンドを使用してキーの wrap-with-trusted 属性を true に設定します。

```
aws-cloudhsm > key set-attribute --filter attr.label=test_key --name wrap-with-
trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

ステップ 2: 信頼できるキーの **CKA_TRUSTED** を true に設定する

キーを信頼できるキーにするには、その CKA_TRUSTED 属性を true に設定する必要があります。これを行うには、CloudHSM CLI または CloudHSM 管理ユーティリティ (CMU) を使用できます。

- CloudHSM CLI を使用してキーの CKA_TRUSTED 属性を設定する場合は、「CloudHSM CLI を使用してキーを信頼できるものとしてマークする」を参照してください。
- CMU を使用してキーの CKA_TRUSTED 属性を設定する場合は、「AWS CloudHSM 管理ユーティリティを使用してキーを信頼済みとしてマークする方法」を参照してください。

ステップ 3. 信頼できるキーを使用してデータキーをラップする

ステップ 1 で参照したデータキーを、ステップ 2 で設定した信頼できるキーでラップするには、以下のリンクにあるコードサンプルを参照してください。それぞれがキーをラップする方法を示しています。

- AWS CloudHSM PKCS #11 の例
- AWS CloudHSM JCE の例

の信頼されたキーを使用してデータキーをラップ解除する方法 AWS CloudHSM

データキーのラップを解除するには AWS CloudHSM、 を true にCKA_UNWRAP設定した信頼されたキーが必要です。このようなキーの条件を満たすには、次の基準も満たしている必要があります。

- キーの CKA_TRUSTED 属性を true に設定する必要があります。
- キーは CKA_UNWRAP_TEMPLATE と関連する属性を使用して、データキーがラップ解除された後に実行できるアクションを指定する必要があります。たとえば、ラップされていないキーをエクスポート不可にしたい場合は、CKA_UNWRAP_TEMPLATE の一部として CKA_EXPORTABLE = FALSE を設定します。

> **Note**
>
> CKA_UNWRAP_TEMPLATE は PKCS #11 でのみ使用できます。

アプリケーションがアンラップするキーを送信するとき、アプリケーションは独自のアンラップテンプレートを提供することもできます。アンラップテンプレートを指定し、アプリケーションが独自のアンラップテンプレートを提供する場合、HSM は両方のテンプレートを使用して属性名と値をキーに適用します。ただし、アンラップリクエスト中に信頼できるキーの CKA_UNWRAP_TEMPLATE の値がアプリケーションによって提供された属性と競合する場合、アンラップリクエストは失敗します。

データキーを信頼できるキーでラップ解除する例については、この PKCS #11 の例を参照してください。

CloudHSM CLI によるキー管理

最新の SDK バージョンシリーズを使用する場合は、CloudHSM CLI を使用して AWS CloudHSM クラスター内のキーを管理します。詳細については、以下のトピックを参照してください。

- 「信頼できるキーの使用」では、CloudHSM CLI を使用してデータを保護するための信頼できるキーを作成する方法について説明します。
- 「キーの生成」には、対称キー、RSA キー、EC キーなどのキーの作成手順が含まれています。
- 「キーの削除」では、キー所有者がキーを削除する方法について説明しています。
- 「キーの共有と共有解除」では、キー所有者がキーを共有および共有解除する方法について詳しく説明しています。

- 「キーをフィルタリングする」には、フィルタを使用してキーを検索する方法に関するガイドラインが記載されています。
- キークォーラム認証の管理 (M/N) では、キーを使用したクォーラム認証を設定し、使用する方法に関するガイドラインが提供されています。

CloudHSM CLI を使用してキーを生成する

キーを生成する前に、CloudHSM CLI を起動し、Crypto User (CU) としてログインする必要があります。HSM でキーを生成するには、生成したいキーと対応するタイプのコマンドを使用します。

トピック

- CloudHSM CLI を使用して対称キーを生成する
- CloudHSM CLI を使用して非対称キーを生成する
- AWS CloudHSM 主要な関連トピック

CloudHSM CLI を使用して対称キーを生成する

にリストされているコマンドCloudHSM CLI の generate-symmetric カテゴリを使用して、対称キーを生成します AWS CloudHSM。利用可能なオプションをすべて確認するには、help key generate-symmetric コマンドを使用します。

AES キーの生成

key generate-symmetric aes コマンドを使用して AES キーを生成します。利用可能なオプションをすべて確認するには、help key generate-symmetric aes コマンドを使用します。

Example

次の例では 32 バイトの AES キーを生成します。

```
aws-cloudhsm > key generate-symmetric aes \
    --label aes-example \
    --key-length-bytes 32
```

引数

<LABEL>

AES キーのユーザー定義ラベルを指定します。

必須: はい

<KEY-LENGTH-BYTES>

キーの長さをバイト単位で指定します。

有効な値:

- 16、24、32

必須: はい

<KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された AES キーに設定するキー属性のスペース区切りリストを指定します (例: `sign=true`)。

サポートされている AWS CloudHSM キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: いいえ

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、key set-attribute を使用します。

デフォルトでは、生成されるキーは永続/トークンキーです。<SESSION> を使用するとこれは変更され、この引数で生成されたキーがセッション/エフェメラルであることが保証されます

必須: いいえ

汎用シークレットキーを生成します

key generate-symmetric generic-secret コマンドを使用して汎用シークレットキーを生成します。利用可能なオプションをすべて確認するには、help key generate-symmetric generic-secret コマンドを使用します。

Example

次の例では、32 バイトの汎用シークレットキーを生成します。

```
aws-cloudhsm > key generate-symmetric generic-secret \
    --label generic-secret-example \
    --key-length-bytes 32
```

引数

<LABEL>

汎用シークレットキーのユーザー定義ラベルを指定します。

必須: はい

<KEY-LENGTH-BYTES>

キーの長さをバイト単位で指定します。

有効な値:

- 1 ~ 800

必須: はい

<KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された汎用シークレットキーに設定するキー属性のスペース区切りリストを指定します (例: `sign=true`)。

サポートされている AWS CloudHSM キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: いいえ

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、key set-attribute を使用します。

デフォルトでは、生成されるキーは永続/トークンキーです。<SESSION> を使用するとこれは変更され、この引数で生成されたキーがセッション/エフェメラルであることが保証されます

必須: いいえ

CloudHSM CLI を使用して非対称キーを生成する

にリストされているコマンドを使用してCloudHSM CLI の generate-asymmetric-pair カテゴリ、AWS CloudHSM クラスターの非対称キーペアを生成します。

RSA キーの生成

key generate-asymmetric-pair rsa コマンドを使用して RSA キーペアを生成します 利用可能なオプションをすべて確認するには、help key generate-asymmetric-pair rsa コマンドを使用します。

Example

次の例では、RSA 2,048 ビットのキーペアが生成されます。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
    --public-exponent 65537 \
    --modulus-size-bits 2048 \
    --public-label rsa-public-example \
    --private-label rsa-private-example
```

引数

<PUBLIC_LABEL>

パブリックキーのユーザー定義ラベルを指定します。

必須: はい

<PRIVATE_LABEL>

プライベートキーのユーザー定義ラベルを指定します。

必須: はい

<MODULUS_SIZE_BITS>

モジュラスの長さをビット単位で指定します。最小値は 2,048 です。

必須: はい

<PUBLIC_EXPONENT>

パブリック指数を指定します。値は、65537 以上の奇数にする必要があります

必須: はい

<PUBLIC_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された RSA パブリックキーに設定するキー属性のスペース区切りリストを指定します (例: `sign=true`)。

サポートされている AWS CloudHSM キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: いいえ

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、key set-attribute を使用します。

デフォルトでは、生成されるキーは永続/トークンキーです。<SESSION> を使用するとこれは変更され、この引数で生成されたキーがセッション/エフェメラルであることが保証されます

必須: いいえ

EC (楕円曲線暗号) キーペアの生成

key generate-asymmetric-pair ec コマンドを使用して EC キーペアを生成します。サポートされている楕円曲線のリストを含め、利用可能なオプションをすべて確認するには、help key generate-asymmetric-pair ec コマンドを使用します。

Example

次の例では、Secp384r1 楕円曲線 を使用して EC キーペアを生成します。

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp384r1 \
```

```
    --public-label ec-public-example \
    --private-label ec-private-example
```

引数

<PUBLIC_LABEL>

パブリックキーのユーザー定義ラベルを指定します。label に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<PRIVATE_LABEL>

プライベートキーのユーザー定義ラベルを指定します。label に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<CURVE>

楕円曲線の識別子を指定します。

有効な値:

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1
- ed25519 (非 FIPS モードの hsm2m.medium インスタンスでのみサポート)

必須: はい

<PUBLIC_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された EC パブリックキーに設定するキー属性のスペース区切りリストを指定します (例: verify=true)。

サポートされている AWS CloudHSM キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: いいえ

<PRIVATE_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された EC プライベートキーに設定するキー属性のスペース区切りリストを指定します (例: `sign=true`)。

サポートされている AWS CloudHSM キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: いいえ

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、key set-attribute を使用します。

デフォルトでは、生成されるキーは永続 (トークン) キーです。<SESSION> で渡すことでこれが変わり、この引数で生成されたキーがセッション (エフェメラル) キーであることが保証されます。

必須: いいえ

ML-DSA キーペアの生成

key generate-asymmetric-pair ml-dsa コマンドを使用して ML-DSA キーペアを生成します。利用可能なオプションをすべて確認するには、help key generate-asymmetric-pair ml-dsa コマンドを使用します。

Example

次の例では、ML-DSA-44 アルゴリズムを使用して ML-DSA キーペアを生成します。

```
aws-cloudhsm > key generate-asymmetric-pair ml-dsa \
    --mldsa-algorithm ML-DSA-44 \
```

```
    --public-label ml-dsa-public-example \
    --private-label ml-dsa-private-example
```

引数

<PUBLIC_LABEL>

パブリックキーのユーザー定義ラベルを指定します。

必須: はい

<PRIVATE_LABEL>

プライベートキーのユーザー定義ラベルを指定します。

必須: はい

<MLDSA_ALGORITHM>

ML-DSA アルゴリズムを指定します。

有効な値:

- ML-DSA-44
- ML-DSA-65
- ML-DSA-87

必須: はい

AWS CloudHSM 主要な関連トピック

AWS CloudHSMのキーに関する追加情報については、次のセクションを参照してください。

- CloudHSM CLI のキー属性
- CloudHSM CLI の generate-asymmetric-pair カテゴリ
- CloudHSM CLI の generate-symmetric カテゴリ

CloudHSM CLI を使用してキーを削除する

このトピックの例を使用して、CloudHSMCLI でキーを削除します。キー所有者のみがキーを削除できます。

トピック

- 例: キーを削除する
- 関連トピック

例: キーを削除する

1. key list コマンドを実行して、削除するキーを特定します。

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_key_to_delete",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
```

```
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
 "0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
```

2. キーを特定したら、キーの固有 label 属性を使用して key delete を実行し、キーを削除します。

```
aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. キーの固有 label 属性を使用して key list コマンドを実行し、キーが削除されたことを確認します。次の例に示すように、HSM クラスターには my_key_to_delete ラベルの付いたキーはありません。

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "matched_keys": [],
```

```
    "total_key_count": 0,
    "returned_key_count": 0
  }
}
```

関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI でキーを削除する

CloudHSM CLI を使用してキーを共有または共有解除する

このトピックのコマンドを使用して、CloudHSMCLI でキーを共有または共有解除します。では AWS CloudHSM、キーを作成する Crypto User (CU) がキーを所有します。所有者は key share と key unshare コマンドを使用してキーを他の CU と共有または共有解除することができます。キーを共有するユーザーは、暗号化オペレーションでキーを使用できますが、キーを削除、エクスポート、共有、共有解除、取得、またはラップすることはできません。

キーを共有する前に、キーを所有する crypto user (CU) として HSM にログインする必要があります。

トピック

- 例: キーの共有と共有解除
- 関連トピック

例: キーの共有と共有解除

Example

次の例は、Crypto User (CU) `alice` でキーを共有および共有解除する方法を示しています。key share コマンドと key unshare コマンドに加えて、コマンドの共有と共有解除には CloudHSM CLI キーフィルタを使用する特定のキーと、キーを共有または共有解除するユーザーの特定のユーザー名も必要です。

1. まず、フィルタを指定して key list コマンドを実行して特定のキーを返し、そのキーがすでに誰と共有されているかを確認します。

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
```

```
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```


2. shared-users 出力を表示して、キーが現在誰と共有されているかを確認します。
3. このキーを Crypto User (CU) alice と共有するには、以下のコマンドを入力します。

```
aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}
```

このコマンドでは、key share コマンドに加えて、キーの固有ラベルと、キーを共有するユーザーの名前が使用されることに注意してください。

4. key list コマンドを実行して、キーが alice と共有されていることを確認します。

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
```

```
            "username": "cu5",
            "key-coverage": "full"
          },
          {
            "username": "cu6",
            "key-coverage": "full"
          },
          {
            "username": "cu7",
            "key-coverage": "full"
          },
          {
            "username": "alice",
            "key-coverage": "full"
          }
        ],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa_key_to_share",
        "id": "",
        "check-value": "0xae8ff0",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": true,
        "verify": false,
```

```
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

5. alice と同じキーを共有解除するには、以下の unshare コマンドを実行します。

```
aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
 attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key unshared successfully"
  }
}
```

このコマンドでは、key unshare コマンドに加えて、キーの固有ラベルと、キーを共有するユーザーの名前が使用されることに注意してください。

6. key list コマンドをもう一度実行して、キーが暗号化ユーザー alice と共有解除されたことを確認します。

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
```

```
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "cu2",
            "key-coverage": "full"
          },
          {
            "username": "cu1",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
          {
            "username": "cu5",
            "key-coverage": "full"
          },
          {
            "username": "cu6",
            "key-coverage": "full"
          },
          {
            "username": "cu7",
            "key-coverage": "full"
          },
        ],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa_key_to_share",
        "id": "",
        "check-value": "0xae8ff0",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
```

```
            "always-sensitive": true,
            "derive": false,
            "destroyable": true,
            "extractable": true,
            "local": true,
            "modifiable": true,
            "never-extractable": false,
            "private": true,
            "sensitive": true,
            "sign": true,
            "trusted": false,
            "unwrap": true,
            "verify": false,
            "wrap": false,
            "wrap-with-trusted": false,
            "key-length-bytes": 1219,
            "public-exponent": "0x010001",
            "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
            "modulus-size-bits": 2048
          }
        }
      ],
      "total_key_count": 1,
      "returned_key_count": 1
    }
}
```


関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーを共有する
- CloudHSM CLI を使用してキーの共有を解除する
- CloudHSM CLI を使用してキーをフィルタリングする

CloudHSM CLI を使用してキーをフィルタリングする

以下のキーコマンドを使用して、CloudHSM CLI の標準化されたキーフィルタリングメカニズムを利用します。

- key list
- key delete
- key share
- key unshare
- key set-attribute

CloudHSM CLI でキーを選択またはフィルタリングするには、キーコマンドは CloudHSM CLI のキー属性 に基づく標準化されたフィルタリングメカニズムを利用します。キーまたはキーのセットは、単一のキーまたは複数のキーを識別できる 1 つ以上の AWS CloudHSM 属性を使用して、キーコマンドで指定できます。キーフィルタリングメカニズムは、現在ログインしているユーザーが所有および共有しているキーと、 AWS CloudHSM クラスター内のすべてのパブリックキーに対してのみ動作します。

トピック

- 要件
- 1 つのキーを検索するためのフィルタリング
- フィルタリングエラー
- 関連トピック

要件

キーをフィルタリングするには、Crypto User (CU) としてログインしている必要があります。

1 つのキーを検索するためのフィルタリング

以下の例では、フィルタとして使用する各属性は `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` の形式で書き込む必要があることに注意してください。例えばラベル属性でフィルタリングする場合は、`attr.label=my_label` と書き込みます。

Example 1 つの属性を使用して 1 つのキーを検索する

この例は、1 つの識別属性のみを使用して一意のキー 1 つにフィルタリングする方法を示しています。

```
aws-cloudhsm > key list --filter attr.label="my_unique_key_label" --verbose
{
```

```
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_unique_key_label",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
```

```
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

Example複数の属性を使用して 1 つのキーを検索する

次の例では、複数のキー属性を使用して 1 つのキーを検索する方法を示します。

```
aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key attr.check-
value=0x29bbd1 --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
```

```
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_crypto_user",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
 "0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```


Exampleフィルタリングしてキーセットを検索する

以下は、プライベート RSA キーのセットを検索するためのフィルタリング方法を示す例です。

```
aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
```

```
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
  "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5
          "modulus-size-bits": 2048
        }
      },
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_test_key",
          "id": "",
          "check-value": "0x29bbd1",
```

```
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
 "0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 2,
    "returned_key_count": 2
  }
}
```

フィルタリングエラー

特定のキー操作は、一度に 1 つのキーに対してしか実行できません。これらの操作では、フィルタリング基準が十分に調整されておらず、複数のキーが条件に一致する場合、CloudHSM CLI はエラーを表示します。その一例として、キー削除を使用した例を以下に示します。

Example一致するキーが多すぎるとフィルタリングエラーが発生します

```
aws-cloudhsm > key delete --filter attr.key-type=rsa
{
  "error_code": 1,
```

```
  "data": "Key selection criteria matched 48 keys. Refine selection criteria to select
 a single key."
}
```

関連トピック

- CloudHSM CLI のキー属性

CloudHSM CLI を使用してキーを信頼できるものとしてマークする

このセクションでは、CloudHSM CLI を使用してキーを信頼できるものとしてマークする方法について説明します。

1. CloudHSM CLI login コマンドを使用して、Crypto User (CU) としてログインします。
2. key list コマンドを使用して、信頼できるとマークしたいキーのキーリファレンスを特定します。次の例では、キーを test_aes_trusted ラベル付きで表示します。

   ```
   aws-cloudhsm > key list --filter attr.label=test_aes_trusted
           {
     "error_code": 0,
     "data": {
       "matched_keys": [
         {
           "key-reference": "0x0000000000200333",
           "attributes": {
             "label": "test_aes_trusted"
           }
         }
       ],
       "total_key_count": 1,
       "returned_key_count": 1
     }
   }
   ```

3. CloudHSM CLI で HSM からのログアウトする コマンドを使用して、Crypto User (CU) としてログインします。
4. CloudHSM CLI を使用して HSM にログインする コマンドを使用して、管理者としてログインします。
5. key set-attribute コマンドと、ステップ 2 で特定したキーリファレンスを指定して、キーの信頼できる値を「true」に設定します。

```
aws-cloudhsm > key set-attribute --filter key-reference=<Key Reference> --name
 trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

CloudHSM CLI を使用したクォーラム認証の管理 (M of N アクセスコントロール)

AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSMs) は、M of N アクセスコントロールとも呼ばれるクォーラム認証をサポートしています。クォーラム認証を使用すると、HSM 上でクォーラムコントロールされたオペレーションを単独のユーザーが実行することはできません。代わりに、HSM ユーザーの最小数 (少なくとも 2 人) が、これらのオペレーションを協力して行う必要があります。クォーラム認証は、複数の HSM ユーザーによる承認を必須とすることで、追加の保護レイヤーを提供します。

クォーラム認証は次のオペレーションを制御できます。

- Crypto User による HSM のキー使用および管理 - キーを使用した署名の作成、キーのラッピング、ラップ解除、共有、共有解除、およびキー属性の設定。

重要な考慮事項

- HSM ユーザーは自分のクォーラムトークンに署名できます。つまり、リクエストするユーザーはクォーラム認証に必要な承認の 1 つを提供できます。
- クォーラム管理されたオペレーションに対して、最小数のクォーラム承認者を選択します。選択できる最小数は 2 で、選択できる最大数は 8 です。
- HSM は最大 1,024 個のクォーラムトークンを保存できます。HSM にすでに 1,024 トークンある場合、新しく作成しようとすると、HSM は期限切れのトークンの 1 つを消去します。デフォルトでは、トークンは作成後 10 分で有効期限が切れます。
- 多要素認証 (MFA) が有効になっている場合、クラスターは、クォーラム認証と MFA に同じキーを使用します。クォーラム認証と MFA の詳細については、CloudHSM CLI を使用して MFA を管理するを参照してください。

- サービスごとに一度にアクティブにできるクォーラムオペレーションは 1 つだけです。同じサービス用に新しいクォーラムトークンを生成する前に、サービスのアクティブなクォーラムトークンを完了または削除する必要があります。詳細については、「サポートされているサービスとタイプ」を参照してください。

次のトピックでは、 AWS CloudHSMでのクォーラム認証についてさらに詳細な情報を提供します。

トピック

- CloudHSM CLI のクォーラム認証プロセス
- CloudHSM CLI によるクォーラム認証でサポートされている AWS CloudHSM サービス名とタイプ
- CloudHSM CLI を使用して AWS CloudHSM crypto-users のクォーラム認証を設定する
- CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたキー管理と使用状況

CloudHSM CLI のクォーラム認証プロセス

次の手順は、CloudHSM CLI のクォーラム認証プロセスの概要を示しています。特定のステップとツールについては、CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたキー管理と使用状況 を参照してください。

1. 各ハードウェアセキュリティモジュール (HSM) ユーザーは、署名のための非対称キーを作成します。これは HSM の外部で行い、キーを適切に保護します。
2. 各 HSM ユーザーは HSM にログインし、署名キーの公開部分 (パブリックキー) を HSM に登録します。
3. HSM ユーザーがクォーラム管理されたオペレーションを実行する場合は、HSM にログインし、クォーラムトークンを取得します。
4. HSM ユーザーは、クォーラムトークンを 1 人または複数の他の HSM ユーザーに付与し、承認を求めます。
5. 他の HSM ユーザーは、キーを使用してクォーラムトークンに暗号で署名することにより承認します。これは HSM の外部で行われます。
6. HSM ユーザーが必要な数の承認を得たら、同じユーザーが HSM にログインし、必要な承認 (署名) をすべて含む署名付きクォーラムトークンファイルを提供して、--approval 引数を指定してクォーラム制御オペレーションを実行します。

7. HSM では、それぞれの署名した人の登録されたパブリックキーを使用して署名を確認します。署名が有効な場合、HSM はトークンを承認し、クォーラム制御されたオペレーションが実行されます。

CloudHSM CLI によるクォーラム認証でサポートされている AWS CloudHSM サービス名とタイプ

管理サービス: クォーラム認証は、ユーザーの作成、ユーザーの削除、ユーザーパスワードの変更、クォーラム値の設定、クォーラム機能と MFA 機能の無効化などの管理者権限を持つサービスに使用されます。

Crypto User サービス: クォーラム認証は、特定のキーに関連する Crypt User の特権サービス (キーによる署名、キーの共有/共有解除)、キーのラッピング/ラップ解除、キー属性の設定など) に使用されます。関連付けられたキーのクォーラム値は、そのキーを生成、インポート、またはラップ解除する際に設定されます。クォーラム値は、そのキーが関連付けられているユーザー数 (キーが共有されているユーザーおよびキー所有者を含む) 以下である必要があります。

各サービスタイプはさらに適格なサービス名に分類されます。このサービス名には、実行可能なクォーラムがサポートする特定のサービスオペレーションのセットが含まれます。

サービス名	サービスタイプ	サービスオペレーション
ユーザー	管理者	• user create • user delete • user change-password • user change-mfa
quorum	管理者	• quorum token-sign set-quorum-value
cluster[1]	管理者	• cluster mtls register-trust-anchor • cluster mtls deregister-trust-anchor • cluster mtls set-enforcement
キーの管理	Crypto User	• キーのラップ

サービス名	サービスタイプ	サービスオペレーション
		• キーのラップ解除 • キーシェア • キー共有解除 • key set-attribute
キーの用途	Crypto User	• キーサイン

[1] クラスターサービスは hsm2m.medium でのみ利用できます

CloudHSM CLI を使用して AWS CloudHSM crypto-users のクォーラム認証を設定する

これらのトピックでは、Crypto User によるクォーラム認証を CloudHSM に設定する方法について説明します。これらの手順は初期セットアップ時に一度だけ実行します。その後のキー管理およびキー使用については、CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたキー管理と使用状況 を参照してください。

トピック

- 前提条件
- ステップ 1. 署名のためのキーの作成と登録
- ステップ 2. キー生成時にキークォーラム値を設定する

前提条件

- CloudHSM CLI に精通していること

ステップ 1. 署名のためのキーの作成と登録

クォーラム認証を使用する場合、各 Crypto User が以下のすべてのステップを実行する必要があります。

トピック

- RSA キーペアの作成
- 登録トークンの作成

- 署名なし登録トークンへ署名する
- HSM でパブリックキーを登録する

RSA キーペアの作成

様々なキーペアを作成、保護する方法があります。次の例では、OpenSSL 使用方法を説明しています。

Example— OpenSSL でプライベートキーを作成する

次の例は、OpenSSL を使用して 2,048 ビットの RSA キーを作成する方法を示しています。この例を使用するには、*<crypto_user1.key>* を、キーの保存先のファイル名に置き換えてください。

```
$ openssl genrsa -out <crypto_user1.key>
Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
```

次に、作成したプライベートキーを使用してパブリックキーを生成します。

Example— OpenSSL でパブリックキーを作成する

以下の例は、OpenSSL を使用して先ほど作成したプライベートキーからパブリックキーを作成する方法を示しています。

```
$ openssl rsa -in crypto_user1.key -outform PEM -pubout -out crypto_user1.pub
writing RSA key
```

登録トークンの作成

トークンを作成し、前のステップで生成したプライベートキーを使用して署名します。

登録トークンの作成

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. quorum token-sign generate コマンドを実行して登録トークンを作成します。

```
aws-cloudhsm > quorum token-sign generate --service registration --token /path/
tokenfile
{
  "error_code": 0,
  "data": {
    "path": "/path/tokenfile"
  }
}
```

3. quorum token-sign generate コマンドは、指定されたファイルパスに登録トークンを生成します。トークンファイルを調査します。

```
$ cat /path/tokenfile
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

トークンファイルは、次のもので構成されます。

- approval_data: base64 でエンコードされランダム化されたデータトークン。raw データが最大 245 バイトを超えないもの。
- unsigned: base64 でエンコードされ、SHA256 ハッシュされた approval_data のトークン。
- signed: OpenSSL で以前に生成された RSA 2,048 ビットのプライベートキーを使用した、署名されていないトークンの base64 でエンコードされた署名付きトークン (署名)。

プライベートキーを使用して署名なしトークンに署名し、プライベートキーへのアクセス権があることを示します。Crypto-user をクォーラムユーザーとして AWS CloudHSM クラスターに登録するには、登録トークンファイルに署名とパブリックキーが完全に入力されている必要があります。

署名なし登録トークンへ署名する

1. base64 でエンコードされた署名なしトークンをデコードし、バイナリファイルに入れます。

```
$ echo -n '6BMUj6mUjjko6ZLCEdzGlWpR5sILhFJfqhW1ej3Oq1g=' | base64 -d >
 crypto_user.bin
```

2. OpenSSL とプライベートキーを使用して現在の署名なしバイナリ登録トークンに署名し、バイナリ署名ファイルを作成します。

```
$ openssl pkeyutl -sign \
-inkey crypto_user1.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in crypto_user.bin \
-out crypto_user.sig.bin
```

3. バイナリ署名を base64 にエンコードします。

```
$ base64 -w0 crypto_user.sig.bin > crypto_user.sig.b64
```

4. base64 でエンコードされた署名をコピーしてトークンファイルに貼り付けます。

```
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": <signed token in base64 encoding>
    }
  ]
}
```

HSM でパブリックキーを登録する

キーを作成した後、crypto-user はパブリックキーを AWS CloudHSM クラスターに登録する必要があります。

1. CloudHSM CLI を起動します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. パブリックキーを登録する Crypto User としてサインインします。

   ```
   aws-cloudhsm > login --username crypto_user1 --role crypto-user
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "crypto_user1",
       "role": "crypto-user"
     }
   }
   ```

3. CloudHSM CLI を使用してユーザーのトークン署名クォーラム戦略を登録する を使用してパブリックキーを登録します。詳細については、次の例を参照するか、または help user change-quorum token-sign register コマンドを使用してください。

   Example– AWS CloudHSM クラスターにパブリックキーを登録する

   以下の例では、CloudHSM CLI で user change-quorum token-sign register コマンドを使用して、Crypto User のパブリックキーを HSM に登録する方法を示しています。このコマンドを使用するには、Crypto User が HSM にログインしている必要があります。以下の値を自分の値に置き換えてください。

   ```
   aws-cloudhsm > user change-quorum token-sign register --public-key </path/
   crypto_user.pub> --signed-token </path/tokenfile>
   {
   ```

```
  "error_code": 0,
  "data": {
    "username": "crypto_user1",
    "role": "crypto-user"
  }
}
```

> **Note**
>
> /path/crypto_user.pub: パブリックキー PEM ファイルへのファイルパス
> 必須: はい
> /path/token_file: ユーザーのプライベートキーによって署名されたトークンを含むファイルパス
> 必須: はい

4. すべての Crypto User がパブリックキーを登録すると、user list コマンドの出力にあるクォーラムフィールドに、有効になっているクォーラム戦略が表示されます。

   この例では、 user list コマンドからの次の出力に示すように、 AWS CloudHSM クラスターには 2 つの HSMs があり、それぞれに同じ crypto-users があります。ユーザー作成の詳細については、CloudHSM CLI によるユーザー管理 を参照してください。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "crypto_user1",
        "role": "crypto-user",
        "locked": "false",
        "mfa": [],
        "quorum": [
```

```
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "crypto_user2",
        "role": "crypto-user",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "crypto_user3",
        "role": "crypto-user",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
```

```
}
```

ステップ 2. キー生成時にキークォーラム値を設定する

クォーラム認証を使用するには、Crypto User が HSM にログインし、関連付けられたキークォーラム値を設定する必要があります。これは、HSM ユーザー管理/使用オペレーションを実行するために必要な Crypto User 承認の最小数です。キー管理またはキーの使用に関連するキーコマンドの詳細については、サポートされているサービスとタイプ を参照してください。

キークォーラム値が設定されたキーペアを生成する

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. CloudHSM CLI を使用して、Crypto User としてログインします。

   ```
   aws-cloudhsm > login --username crypto_user1 --role crypto-user
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "crypto_user1",
       "role": "crypto-user"
     }
   }
   ```

この例では、キー管理オペレーションとキー使用オペレーションの両方に対してクォーラム値 2 が設定された RSA キーペアを生成します。クォーラム値として設定できるのは、0 ~ 8 の範囲で、HSM 上の Crypto User の総数まで選択できます。この例では、HSM には 3 名の Crypto User が存在するため、設定可能な最大値は 3 になります。この例ではキー生成時に *<crypto_user2>* とキーを共有している点に注意してください。さらに、パブリックキーにはクォーラム値は存在しないことにも注意が必要です。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes verify=true \
--private-attributes sign=true
--share-crypto-users crypto_user2 \
--manage-private-key-quorum-value 2 \
--use-private-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000640006",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto_user",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "0x",
        "check-value": "0x218f50",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xbdf471a3d2a869492f51c767bece8780730ae6479a9a75efffe7cea3594fb28ca518630e7b1d988b45d2fedc830
        "modulus-size-bits": 2048
      }
    },
    "private_key": {
      "key-reference": "0x0000000000640007",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto_user",
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "crypto_user2",
            "key-coverage": "full"
          }
        ],
        "key-quorum-values": {
          "manage-key-quorum-value": 2,
          "use-key-quorum-value": 2
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "0x",
        "check-value": "0x218f50",
        "class": "private-key",
```

```
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1216,
        "public-exponent": "0x010001",
        "modulus":
 "0xbdf471a3d2a869492f51c767bece8780730ae6479a9a75efffe7cea3594fb28ca518630e7b1d988b45d2fedc830
        "modulus-size-bits": 2048
      }
    }
  }
}
```


クォーラム制御を使用してキーを生成する場合、キーには、キーの最大クォーラム値と同じ数以上のユーザーを関連付ける必要があります。関連付けられるユーザーには、キー所有者と、そのキーが共有されている Crypto User が含まれます。キーを共有すべき最小ユーザー数を決定するには、キー使用クォーラム値とキー管理クォーラム値のうち大きい方のクォーラム値を取得し、デフォルトでキーに関連付けられているキー所有者を考慮して 1 を差し引きます。キーをさらに多くのユーザーと共有するには、CloudHSM CLI を使用してキーを共有する コマンドを使用します。

キー生成時に、必要な人数のユーザーとキーを共有しないと、以下に示すように失敗します。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes verify=true \
```

```
--private-attributes sign=true
--share-crypto-users crypto_user2 crypto_user3 \
--manage-private-key-quorum-value 3 \
--use-private-key-quorum-value 4
{
  "error_code": 1,
  "data": "Invalid quorum value provided."
}
```

CloudHSM CLI AWS CloudHSM を使用するためのクォーラム認証を有効にしたキー管理と使用状況

AWS CloudHSM クラスターのクォーラム認証を設定した後、キーにクォーラム値が関連付けられている場合、暗号化ユーザーは HSM キー管理または使用操作を単独で実行できません。本トピックでは、Crypto User が HSM のキー管理オペレーションまたはキー使用オペレーションを実行するための一時的なトークンを取得する方法について説明します。

> **Note**
>
> 各クォーラムトークンは 1 回のオペレーションで有効です。そのオペレーションが成功すると、トークンは無効となり、Crypto User は新しいトークンを取得する必要があります。クォーラムトークンは、現在のログインセッション中のみ有効です。CloudHSM CLI からログアウトするか、ネットワークが切断された場合、トークンは無効になり、新しいトークンを取得する必要があります。CloudHSM トークンは CloudHSM CLI 内でのみ使用できます。別のアプリケーションの認証に使用することはできません。

次の例は、クォーラム認証が設定された後、Crypto User がクォーラムが関連付けられたキーを使用して HSM 上で署名を作成しようとした際の出力例です。このコマンドは `Quorum Failed` エラーで失敗します。これは、クォーラム認証が失敗したことを意味します。

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private-key-example --
hash-function sha256 --data YWJjMTIz
{
  "error_code": 1,
  "data": "Quorum Failed"
}
```

Crypto User が HSM 上でキー管理オペレーションまたはキー使用オペレーションを実行するための一時的なトークンを取得するには、次の作業を完了する必要があります。

Steps

- ステップ 1. クォーラムトークンの取得
- ステップ 2. 承認する Crypto User から署名を取得する
- ステップ 3. CloudHSM クラスターでトークンを承認し、オペレーションを実行する

ステップ 1. クォーラムトークンの取得

1. CloudHSM CLI を起動します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. Crypto User としてクラスターにログインします。

   ```
   aws-cloudhsm > login --username <crypto_user1> --role crypto-user --
   password password123
   ```

   この例では、crypto_user1 を crypto-user ロールで CloudHSM CLI にサインインさせています。以下の値を自分の値に置き換えてください。

   ```
   {
     "error_code": 0,
     "data": {
       "username": "crypto_user1",
       "role": "crypto-user"
     }
   }
   ```

3. quorum token-sign generate コマンドを使用してクォーラムトークンを生成します。

   次のコマンドでは、key-usage は生成するトークンを使用するサービス名を示します。この場合、トークンは key-usage オペレーション (key-usage サービス) 用です。この例では、--filter フラグを使用して、特定のキーにトークンを関連付けています。

```
aws-cloudhsm > quorum token-sign generate --service key-usage --token </path/
crypto_user1.token> --filter attr.label=rsa-private-key-example
{
  "error_code": 0,
  "data": {
    "path": "/home/crypto_user1.token"
  }
}
```

この例では、ユーザー名 crypto_user1 の Crypto User のクォーラムトークンを取得し、そのトークンを crypto_user1.token という名前のファイルに保存します。この例のコマンドを使用するには、以下の値を独自のものに置き換えてください。

quorum token-sign generate コマンドは、指定されたファイルパスでキー使用サービスのクォーラムトークンを生成します。トークンファイルは以下のように検査できます。

```
$ cat </path/crypto_user1.token>
{
  "version": "2.0",
  "service": "key-usage",
  "key_reference": "0x0000000000680006",
  "approval_data":
 "AAIABQAAABkAAAAAGgABi5CDa9x9VyyRIaFbkSrHgJjcnlwdG9fdXNlcgAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+GJj8gXo9lKuANGNyeXB0b191c2VyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGc9AEsAAAAAAAAAAA==",
  "token": "5GlgoWOlQU4fw4QIlbxkPGZVOVoDugFGuSKE/k67ncM=",
  "signatures": []
}
```

トークンファイルは、次のもので構成されます。

- service: トークンが関連付けられているクォーラムサービスの識別子。
- key_reference: このクォーラムトークンが関連付けられているキーの識別子。
- approval_data: HSM によって生成された base64 でエンコードされた raw データトークン。
- token: base64 でエンコードされ、SHA-256 ハッシュされた approval_data のトークン
- signatures: 署名なしトークンに対する Base64 エンコードされた署名付きトークン (署名) の配列。各承認者の署名は、次の形式の JSON オブジェクトリテラルで表されます。

```
{
      "username": "<APPROVER_USERNAME>",
```

```
      "role": "<APPROVER_ROLE>",
      "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
}
```

各署名は、パブリックキーが HSM に登録されている承認者が、対応する RSA 2,048 ビットプライベートキーを使用した結果から作成されます。

4. 新しいユーザーサービスのクォーラムトークンを検証します。quorum token-sign list コマンドを実行すると、そのトークンが CloudHSM 上に存在することが確認できます。

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "crypto_user",
        "service": "key-usage",
        "key-reference": "0x0000000000680006",
        "minimum-token-count": 2
      }
    ]
  }
}
```

`minimum-token-count` は、クラスター内の 1 台の HSM から取得される、ユーザー名、サービス、key-reference に対応するキートークンの使用可能な最小数をクラスター全体として集約したビューを示します。

たとえば、2 台構成の HSM クラスターを想定します。クラスター内の一方の HSM から、ユーザー `crypto_user1` によりキー参照 `0x0000000000680006` に対して生成された key-usage トークンを 2 個取得し、さらにもう一方の HSM から、ユーザー `crypto_user1` によりキー参照 `0x0000000000680006` に対して生成された key-usage トークンを 1 個取得したとします。この場合、`"minimum-token-count": 1` が表示されます。

ステップ 2. 承認する Crypto User から署名を取得する

クォーラムトークンを保持している Crypto User は、他の Crypto User からそのトークンの承認を得る必要があります。承認を与える Crypto User は、自身の署名キーを使用して、HSM の外部でトークンに対して暗号学的署名を行います。

トークンの署名にはさまざまな方法が使用されます。次の例は、OpenSSL を使用してトークンに署名する方法を示しています。別の署名ツールを使用する場合は、そのツールが Crypto User のプライベートキー (署名キー) を使用して、トークンの SHA-256 ダイジェストに署名することを必ず確認してください。

この例では、トークン (`crypto-user`) を持つ Crypto User に少なくとも 2 つの承認が必要です。以下のコマンド例では、2 人の Crypto User が OpenSSL を使用してトークンに暗号で署名する方法を示します。

1. base64 でエンコードされた署名なしトークンをデコードし、バイナリファイルに入れます。

```
$echo -n '5GlgoWOlQU4fw4QIlbxkPGZVOVoDugFGuSKE/k67ncM=' | base64 -d >
 crypto_user1.bin
```

2. ユーザーサービス用のバイナリ形式のクォーラム未署名トークンに対して、承認者のプライベートキーを使用して OpenSSL で署名し、バイナリ署名ファイルを作成します。

```
$openssl pkeyutl -sign \
-inkey crypto_user1.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in crypto_user1.bin \
-out crypto_user1.sig.bin
```

3. バイナリ署名を base64 にエンコードします。

```
$ base64 -w0 crypto_user1.sig.bin > crypto_user1.sig.b64
```

4. 承認者署名用に以前に指定した JSON オブジェクトリテラル形式を使用して、base64 でエンコードされた署名をコピーしてトークンファイルに貼り付けます。

```
{
  "version": "2.0",
  "service": "key-usage",
  "key_reference": "0x0000000000680006",
  "approval_data":
 "AAIABQAAABkAAAAAGgABi5CDa9x9VyyRIaFbkSrHgJjcnlwdG9fdXNlcgAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+GJj8gXo9lKuANGNyeXB0b191c2VyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGc9AEsAAAAAAAAAAA==",
  "token": "5GlgoWOlQU4fw4QIlbxkPGZVOVoDugFGuSKE/k67ncM=",
  "signatures": [
    {
      "username": "crypto_user1",
```

```
      "role": "crypto-user",
      "signature": "wa7aPzmGwBjcEoZ6jAzYASp841AfgOvcI27Y/
tGlCj1E9DawnFw5Uf0IJT2Ca7T5XD2ThVkUi0B+dhAomdqYNl6aUUFrJyH9GBJ
+E0PmA5jNVm25tzeRWBJzneTg4/
zTeE2reNqrHFHicWnttQLe9jS09J1znuDGWDe0HaBKWUaz2gUInJRqmeXDsZYdSvZksrqUH5dci/
RsaDE2+tGiS9g0RcIkFbsPW4HpGe2e5HVzGsqrV8O3PKlYQv6+fymfcNTTuoxKcHAkOjpl43QSuSIu2gVq7KI8mSmmW
+oiukaNfLJr+MoDKzAvCGDg4cDArg=="
    },
    {
      "username": "crypto_user2",
      "role": "crypto-user",
      "signature": "wa7aPzmGwBjcEoZ6jAzYASp841AfgOvcI27Y/
tGlCj1E9DawnFw5Uf0IJT2Ca7T5XD2ThVkUi0B+dhAomdqYNl6aUUFrJyH9GBJ
+E0PmA5jNVm25tzeRWBJzneTg4/
zTeE2reNqrHFHicWnttQLe9jS09J1znuDGWDe0HaBKWUaz2gUInJRqmeXDsZYdSvZksrqUH5dci/
RsaDE2+tGiS9g0RcIkFbsPW4HpGe2e5HVzGsqrV8O3PKlYQv6+fymfcNTTuoxKcHAkOjpl43QSuSIu2gVq7KI8mSmmW
+oiukaNfLJr+MoDKzAvCGDg4cDArg=="
    }
  ]
}
```

ステップ 3. CloudHSM クラスターでトークンを承認し、オペレーションを実行する

Crypto User が必要な承認および署名を取得すると、そのトークンをキー管理オペレーションまたはキー使用オペレーションとともに CloudHSM クラスターに提供できます。

キーのオペレーションが、クォーラムトークンに関連付けられた適切なクォーラムサービスに対応していることを必ず確認してください。詳細については、サポートされているサービスとタイプ を参照してください。

トランザクション中、トークンは AWS CloudHSM クラスター内で承認され、リクエストされたキーオペレーションを実行します。キーオペレーションが成功するには、承認済みで有効なクォーラムトークンと有効なキーオペレーションの両方が必要です。

Example RSA-PKCS メカニズムを使用して署名を生成する

次の例では、ログイン中の Crypto User が、HSM 上のキーを使用して署名を作成します。

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private-key-example --
hash-function sha256 --data YWJjMTIz --approval /path/crypto_user1.token

{
```

```
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000640007",
    "signature":
 "h6hMqXacBrT3x3MXV13RXHdQno0+IQ6iy0kVrGzo23+eoWT0ZZgrSpBCu5KcuP6IYYHw9goQ5CfPf4jI1nO5m/
IUJtF1A1lmcz0HjEy1CJ7ICXNReDRyeOU8m43dkJzt0OUdkbtkDJGAcxkbKHLZ02uWsGXaQ8bOKhoGwsRAHHF6nldTXquIC
+pZmUS38ythybney94Wj6fzYOER8v7VIY5ijQGa3LfxrjSG4aw6QijEEbno5LSf18ahEaVKmVEnDBL54tylCJBGvGsYSY9H
TDd2wfvP4PaxbFRyyHaw=="
  }
}
```

Crypto User が同じトークンを使ってもう一度 HSM のキー使用オペレーションを試みると、その操作は失敗します。

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private-key-example --
hash-function sha256 --data YWJjMTIz --approval /home/crypto_user1.token
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

別の HSM キーオペレーションを実行するには、Crypto User は新しいクォーラムトークンを生成し、承認者から新しい署名を取得し、--approval 引数でクォーラムトークンを指定して目的のキーオペレーションを実行する必要があります。

利用可能なトークンを確認するには、quorum token-sign list を使用します。次の例では、Crypto User が承認済みトークンを持っていないことが示されています。

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}
```

KMU AWS CloudHSM によるキー管理

最新の SDK バージョンシリーズを使用する場合は、CloudHSM CLI を使用して AWS CloudHSM クラスター内のキーを管理します。

以前の SDK バージョンシリーズを使用している場合は、key_mgmt_util (KMU) コマンドラインツールを使用して、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) のキーを管理できます。キーを管理する前に、 AWS CloudHSM クライアントを起動し、key_mgmt_util を起動して、HSMs に ログインする必要があります。さらなる詳細については、Getting Started with key_mgmt_util を参照してください。

- 「Using trusted keys」では、PKCS #11 ライブラリ属性と CMU を使用してデータを保護するための信頼できるキーを作成する方法について説明しています。
- 「キーの生成」には、対称キー、RSA キー、EC キーなどのキーの生成手順が含まれています。
- 「キーのインポート」には、キー所有者がキーをインポートする方法の詳細が記載されています。
- 「キーのエクスポート」には、キー所有者がキーをエクスポートする方法の詳細が記載されています。
- 「キーの削除」では、キー所有者がキーを削除する方法の詳細が記載されています。
- 「キーの共有と共有解除」では、キー所有者がキーを共有および共有解除する方法について詳しく説明しています。

KMU AWS CloudHSM を使用してキーを生成する

ハードウェアセキュリティモジュール (HSM) でキーを生成するには、生成するキーのタイプに対応する AWS CloudHSM key_mgmt_util (KMU) の コマンドを使用します。

トピック

- AWS CloudHSM KMU を使用して対称キーを生成する
- KMU を使用して RSA AWS CloudHSM キーペアを生成する
- KMU を使用して ECC (楕円曲線暗号化) AWS CloudHSM キーペアを生成する

AWS CloudHSM KMU を使用して対称キーを生成する

AWS CloudHSM key_mgmt_util (KMU) の genSymKey コマンドを使用して、AES やその他のタイプの対称キーを生成します AWS CloudHSM。利用可能なオプションをすべて確認するには、genSymKey -h コマンドを使用します。

以下の例では、256 ビット AESキーが作成されます。

```
Command: genSymKey -t 31 -s 32 -l aes256
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS
```

```
Symmetric Key Created.  Key Handle: 524295

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

KMU を使用して RSA AWS CloudHSM キーペアを生成する

の RSA キーペアを生成するには AWS CloudHSM、 AWS CloudHSM key_mgmt_util の genRSAKeyPair コマンドを使用します。利用可能なオプションをすべて確認するには、genRSAKeyPair -h コマンドを使用します。

次の例では、RSA 2,048 ビットのキーペアが生成されます。

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa2048
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 524294    private key handle: 524296

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

KMU を使用して ECC (楕円曲線暗号化) AWS CloudHSM キーペアを生成する

の ECC キーペアを生成するには AWS CloudHSM、 AWS CloudHSM key_mgmt_util の genECCKeyPair コマンドを使用します。サポートされている楕円曲線のリストを含め、利用可能なオプションをすべて確認するには、genECCKeyPair -h コマンドを使用します。

次の例では、NIST FIPS publication 186-4 で定義されている P-384 楕円曲線を使用して ECC キーペアを生成します。

```
Command: genECCKeyPair -i 14 -l ecc-p384
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 524297    private key handle: 524298

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

KMU AWS CloudHSM を使用してキーをインポートする

AWS CloudHSM key_mgmt_util を使用してシークレットキー、つまり対称キーと非対称プライベートキーをハードウェアセキュリティモジュール (HSM) にインポートするには、まず HSM にラッピングキーを作成する必要があります。ラップキーなしで直接パブリックキーをインポートすることができます。

トピック

- KMU AWS CloudHSM を使用してシークレットキーをインポートする
- KMU AWS CloudHSM を使用してパブリックキーをインポートする

KMU AWS CloudHSM を使用してシークレットキーをインポートする

key_mgmt_util (KMU) AWS CloudHSM を使用してシークレットキーを にインポートするには、次の手順を実行します。シークレットキーをインポートする前に、ファイルに保存します。対称キーを raw バイトとして保存し、非対称プライベートキーを PEM 形式で保存します。

この例では、ファイルからプレーンテキストのシークレットキーを HSM にインポートする方法を示します。暗号化されたキーをファイルから HSM にインポートするには、unWrapKey コマンドを使用します。

シークレットキーをインポートするには

1. genSymKey コマンドを使用してラップキーを作成します。次のコマンドは、現在のセッション中のみ有効な 128 ビット AES ラップキーを作成します。セッションキーまたは永続キーをラップキーとして使用できます。

   ```
   Command: genSymKey -t 31 -s 16 -sess -l import-wrapping-key
   Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

   Symmetric Key Created.  Key Handle: 524299

   Cluster Error Status
   Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
   ```

2. インポートするシークレットキーのタイプに応じて、次のいずれかのコマンドを使用します。

- 対称キーをインポートするには、imSymKey コマンドを使用します。次のコマンドは、前のステップで作成したラップキーを使って aes256.key という名前のファイルから AES キーをインポートします。利用可能なオプションをすべて確認するには、imSymKey -h コマンドを使用します。

```
Command: imSymKey -f aes256.key -t 31 -l aes256-imported -w 524299
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Unwrapped.  Key Handle: 524300

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

- 非対称プライベートキーをインポートするには、importPrivateKey コマンドを使用します。次のコマンドは、前のステップで作成したラップキーを使って rsa2048.key という名前のファイルからプライベートキーをインポートします。利用可能なオプションをすべて確認するには、importPrivateKey -h コマンドを使用します。

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299
BER encoded key length is 1216

Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Private Key Unwrapped.  Key Handle: 524301

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

KMU AWS CloudHSM を使用してパブリックキーをインポートする

AWS CloudHSM key_mgmt_util (KMU) の importPubKey コマンドを使用して、公開キーをインポートします。利用可能なオプションをすべて確認するには、importPubKey -h コマンドを使用します。

次の例では、`rsa2048.pub` という名前のファイルから RSA パブリックキーをインポートします。

```
Command: importPubKey -f rsa2048.pub -l rsa2048-public-imported
Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS

Public Key Handle: 524302

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

KMU AWS CloudHSM を使用してキーをエクスポートする

AWS CloudHSM key_mgmt_util (KMU) を使用してハードウェアセキュリティモジュール (HSM) からシーク AWS CloudHSM レットキー、つまり対称キーと非対称プライベートキーをエクスポートするには、まずラッピングキーを作成する必要があります。ラップキーなしで直接パブリックキーをエクスポートすることができます。

キーをエクスポートできるのは、キー所有者のみです。キーを共有するユーザーは、キーを暗号化オペレーションで使用することはできますが、エクスポートすることはできません。この例を実行する際は、必ず作成したキーをエクスポートします。

> **Important**
>
> exSymKey コマンドは、シークレットキーのプレーンテキストの (暗号化されていない) コピーをファイルにコピーします。エクスポートプロセスではラップキーが必要ですが、ファイルにあるキーはラップされたキーではありません。キーのラップ (暗号化) されたコピーをエクスポートするには、wrapKey コマンドを使用します。

トピック

- KMU AWS CloudHSM を使用してシークレットキーをエクスポートする
- KMU AWS CloudHSM を使用してパブリックキーをエクスポートする

KMU AWS CloudHSM を使用してシークレットキーをエクスポートする

key_mgmt_util (KMU) AWS CloudHSM を使用して からシークレットキーをエクスポートするには、次の手順を実行します。

シークレットキーをエクスポートするには

1. genSymKey コマンドを使用してラップキーを作成します。次のコマンドは、現在のセッション中のみ有効な 128 ビット AES ラップキーを作成します。

```
Command: genSymKey -t 31 -s 16 -sess -l export-wrapping-key
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524304

Cluster Error Status
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

2. エクスポートするシークレットキーのタイプに応じて、次のいずれかのコマンドを使用します。

   - 対称キーをエクスポートするには、exSymKey コマンドを使用します。次のコマンドでは、aes256.key.exp という名前のファイルに AES キーをエクスポートします。利用可能なオプションをすべて確認するには、exSymKey -h コマンドを使用します。

```
Command: exSymKey -k 524295 -out aes256.key.exp -w 524304
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "aes256.key.exp"
```

> **Note**
>
> コマンドの出力には、「ラップされた対称キー」が出力ファイルに書かれている、とあります。ただし、出力ファイルにはプレーンテキストの (ラップされていない) キーが含まれています。ファイルにラップ (暗号化) されたキーをエクスポートするには、wrapKey コマンドを使用します。

- プライベートキーをエクスポートするには、exportPrivateKey コマンドを使用します。次のコマンドでは、rsa2048.key.exp という名前のファイルにプライベートキーをエクスポートします。利用可能なオプションをすべて確認するには、exportPrivateKey -h コマンドを使用します。

```
Command: exportPrivateKey -k 524296 -out rsa2048.key.exp -w 524304
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS

PEM formatted private key is written to rsa2048.key.exp
```

KMU AWS CloudHSM を使用してパブリックキーをエクスポートする

key AWS CloudHSM _mgmt_util (KMU) の exportPubKey コマンドを使用して、パブリックキーをエクスポートします。利用可能なオプションをすべて確認するには、exportPubKey -h コマンドを使用します。

次の例では、rsa2048.pub.exp という名前のファイルに RSA パブリックキーをエクスポートします。

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp
PEM formatted public key is written to rsa2048.pub.key

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

KMU と CMU でキーを削除する

次の例のように deleteKey コマンドを使用てキーを削除します。キーを削除できるのは、キー所有者のみです。

```
Command: deleteKey -k 524300
Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

KMU および CMU とキーを共有および共有解除する

では AWS CloudHSM、キーを作成する CU がキーを所有します。所有者はキーを管理し、エクスポートおよび削除し、また、暗号化オペレーションでキーを使用できます。また、所有者は、他の CU ユーザーとキーを共有することもできます。キーを共有するユーザーは、暗号化オペレーションでキーを使用することはできますが、そのキーのエクスポート、削除、または他のユーザーとの共有はできません。

例えば、genSymKey あるいは genRSAKeyPair コマンドの -u のパラメータを使うことによって、キーを作成するときは、キーを他の CU ユーザーと共有するとができます。別の HSM ユーザーと既存のキーを共有するには、cloudhsm_mgmt_util コマンドラインツールを使用します。これは、このセクションで説明しているほとんどのタスク (key_mgmt_util コマンドラインツールを使用するもの) とは異なります。

キーを共有するには、cloudhsm_mgmt_util を起動し、エンドツーエンドの暗号化を有効にして、HSM にログインする必要があります。キーを共有するには、キーを所有する Crypto User (CU) として HSM にログインします。キーの所有者のみがキーを共有することができます。

shareKey コマンドを使用してキーを共有または共有解除します。キーのハンドルと、ユーザーの ID を指定します。複数のユーザーと共有または共有解除するには、ユーザー ID のカンマ区切りのリストを指定します。キーを共有するには、次の例のように、コマンドの最後のパラメーターとして 1 を使用します。共有解除するには、0 を使用します。

```
aws-cloudhsm > shareKey 524295 4 1
*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)? y
shareKey success on server 0(10.0.2.9)
shareKey success on server 1(10.0.3.11)
shareKey success on server 2(10.0.1.12)
```

shareKey コマンドの構文を次に示します。

```
aws-cloudhsm > shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

AWS CloudHSM 管理ユーティリティを使用してキーを信頼済みとしてマークする方法

このセクションのコンテンツでは、 AWS CloudHSM 管理ユーティリティ (CMU) を使用してキーを信頼済みとしてマークする手順について説明します。

1. loginHSM コマンドを使用して、Crypto Officer (CO) としてログインします。
2. OBJ_ATTR_TRUSTED (value 134) を true (1) に設定して CMU を使用して AWS CloudHSM キーの属性を設定する コマンドを使用します。

```
aws-cloudhsm > setAttribute <Key Handle> 134 1
```

でのクラスターバックアップ AWS CloudHSM

AWS CloudHSM は、クラスターの定期的なバックアップを少なくとも 24 時間に 1 回実行します。各バックアップは、次のデータの暗号化されたコピーを含んでいます。

- ユーザー (CO、CU、および AU)
- キーマテリアルと証明書
- ハードウェアセキュリティモジュール (HSM) の設定とポリシー

バックアップを作成するようサービスに指示することはできませんが、特定の操作を行うことで強制的にバックアップを作成させることは可能です。以下のいずれかのアクションを実行すると、サービスがバックアップを作成します。

- クラスターをアクティベートするには
- HSM をアクティブクラスターに追加します。
- アクティブクラスターから HSM を削除

AWS CloudHSM は、クラスターの作成時に設定したバックアップ保持ポリシーに基づいてバックアップを削除します。バックアップ保持ポリシーの管理については、「バックアップ保持の設定」を参照してください。

トピック

- AWS CloudHSM クラスターバックアップの使用
- AWS CloudHSM クラスターバックアップの削除
- AWS CloudHSM バックアップの復元
- AWS CloudHSM バックアップ保持ポリシーを設定する
- AWS リージョン間での AWS CloudHSM クラスターバックアップのコピー
- での共有バックアップの使用 AWS CloudHSM

AWS CloudHSM クラスターバックアップの使用

以前に 1 つ以上のアクティブな HSMs が含まれ AWS CloudHSM ていた のクラスターにハードウェアセキュリティモジュール (HSM) を追加すると、サービスは最新のバックアップを新しい HSM に復元します。バックアップを使用して、使用頻度が少ない HSM を管理します。HSM が不要な場合

は、それを削除してバックアップをトリガーします。後に、その HSM が必要になったときに、同じクラスター内に新しい HSM を作成します。この操作により、以前に HSM の削除操作で作成したバックアップが復元されます。

有効期限切れのキー、または非アクティブなユーザーの削除

有効期限切れのキー、または非アクティブなユーザーなど、特定の暗号化マテリアルを環境から削除できます。これは 2 ステッププロセスです。まず、HSM からこれらのマテリアルを削除します。次に、既存のバックアップをすべて削除します。このプロセスに従うと、バックアップから新しいクラスターを初期化するときに、削除された情報が復元されないようになります。詳細については、「the section called “バックアップの削除”」を参照してください。

ディザスタリカバリの検討

バックアップからクラスターを作成することができます。これは、クラスターのリカバリ・ポイントを設定するために行う場合があります。リカバリ・ポイントに必要なすべてのユーザー、キー・マテリアル、証明書を含むバックアップを指定し、そのバックアップを使用して新しいクラスターを作成します。バックアップからクラスターを作成する方法の詳細については、バックアップからクラスターを作成する を参照してください。

また、クラスターのバックアップを別のリージョンにコピーして、そこで元のクラスターのクローンとして新しいクラスターを作成することもできます。このプロセスは、さまざまな理由 (例: ディザスタリカバリプロセスの簡素化) で使用できます。バックアップをリージョンにコピーする方法の詳細については、リージョン間のバックアップのコピー を参照してください。

AWS CloudHSM クラスターバックアップの削除

AWS CloudHSM クラスターバックアップを削除すると、サービスはバックアップを 7 日間保持し、その間にバックアップを復元できます。7 日間の期間を過ぎると、バックアップを復元することはできなくなります。バックアップの管理の詳細については、クラスターのバックアップ を参照してください。

次の表では、バックアップを削除する方法を説明します。

Console

バックアップを削除するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。

2. AWS リージョンを変更するには、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、[バックアップ] を選択します。
4. 削除するバックアップを選択します。
5. 選択したバックアップを削除するには、[アクション]、[削除] を選択します。

   [バックアップの削除] ダイアログボックスが表示されます。
6. [削除] を選択します。

   バックアップの状態が PENDING_DELETE に変わります。削除をリクエストしてから最大 7 日間は、削除を保留しているバックアップを復元することができます。

バックアップを一覧表示する方法 (AWS CLI)

- PENDING_DELETION の状態にあるすべてのバックアップのリストを表示するには、describe-backups コマンドを実行し、フィルタとして `states=PENDING_DELETION` を含めます。

```
$ aws cloudhsmv2 describe-backups --filters states=PENDING_DELETION
{
    "Backups": [
        {
            "BackupId": "backup-ro5c4er4aac",
            "BackupState": "PENDING_DELETION",
            "ClusterId": "cluster-dygnwhmscg5",
            "CreateTimestamp": 1534461854.64,
            "DeleteTimestamp": 1536339805.522,
            "HsmType": "hsm2m.medium",
            "Mode": "NON_FIPS",
            "NeverExpires": false,
            "TagList": []
        }
}
```

AWS CLI

バックアップのステータスを確認するか、またはバックアップの ID を検索するには describe-backups コマンドを AWS CLIから実行します。

バックアップを削除するには (AWS CLI)

- コマンドプロンプトで、delete-backup コマンドを実行し、削除するバックアップの ID を渡します。

```
$ aws cloudhsmv2 delete-backup --backup-id <backup ID>
{
    "Backup": {
        "CreateTimestamp": 1534461854.64,
        "ClusterId": "cluster-dygnwhmscg5",
        "BackupId": "backup-ro5c4er4aac",
        "BackupState": "PENDING_DELETION",
        "DeleteTimestamp": 1536339805.522,
        "HsmType": "hsm1.medium",
        "Mode": "FIPS"
    }
}
```

AWS CloudHSM API

API を使用してバックアップを削除する方法については、「DeleteBackup」を参照してください。

AWS CloudHSM バックアップの復元

AWS CloudHSM は削除されたバックアップを 7 日間保持し、その間にバックアップを復元できます。7 日間の期間を過ぎると、バックアップを復元することはできなくなります。バックアップの管理の詳細については、クラスターのバックアップ を参照してください。

次の表では、バックアップを削除する方法を説明します。

Console

バックアップを復元するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. AWS リージョンを変更するには、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、[バックアップ] を選択します。
4. PENDING_DELETE 復元する状態のバックアップを選択します。
5. 選択したバックアップを復元するには、[アクション]、[復元] を選択します。

AWS CLI

バックアップを復元する方法 (AWS CLI)

- バックアップを復元するには、restore-backup コマンドを発行し、PENDING_DELETION 状態のバックアップの ID を渡します。

```
$ aws cloudhsmv2 restore-backup --backup-id <backup ID>
{
    "Backup": {
        "ClusterId": "cluster-dygnwhmscg5",
        "CreateTimestamp": 1534461854.64,
        "BackupState": "READY",
        "BackupId": "backup-ro5c4er4aac"
    }
}
```

AWS CloudHSM API

API を使用してバックアップを復元する方法については、「RestoreBackup」を参照してください。

AWS CloudHSM バックアップ保持ポリシーを設定する

AWS CloudHSM は、クラスターの作成時に設定したバックアップ保持ポリシーに基づいてバックアップを消去します。バックアップ保持ポリシーは、クラスターに適用されます。バックアップを別のリージョンに移動すると、そのバックアップはクラスターに関連付けられなくなり、バックアップ保持ポリシーもなくなります。クラスターに関連付けられていないバックアップは手動で削除する必要があります。 AWS CloudHSM はクラスターの最後のバックアップを削除しません。hsm1.medium から移行されたクラスターの場合、hsm1.medium がサポート終了になるまで、は最新の hsm1.medium バックアップ AWS CloudHSM を保持します。

AWS CloudTrail は、削除の対象となるバックアップを報告します。サービスが削除したバックアップは、手動で削除したバックアップ を復元する場合と同様に、復元できます。競合状態を回避する

には、サービスによって削除されたバックアップを復元する前に、クラスターのバックアップ保持ポリシーを変更する必要があります。保持ポリシーを変更せず、選択したバックアップを保持したい場合は、クラスターのバックアップ保持ポリシーから バックアップを除外する ように指定できます。

AWS CloudHSM 料金の詳細については、「」を参照してくださいニーズに合わせてスケールすることでコストを削減。

マネージドバックアップの保持

2020 年 11 月 18 日より前に作成されたクラスターには、90 日とクラスタの経過時間を足したバックアップ保持ポリシーが適用されます。例えば、2019 年 11 月 18 日にクラスターを作成した場合、サービスはクラスターに 1 年間と 90 日 (455 日) のバックアップ保持ポリシーを割り当てます。この期間は 7～379 日間の任意の数に設定できます。 AWS CloudHSM はクラスターの最後のバックアップを削除しません。バックアップの管理の詳細については、クラスターのバックアップ を参照してください。

> **Note**
>
> AWS サポート に連絡することで、マネージドバックアップ保持を完全にオプトアウトできます。

次の表は、バックアップ保持を設定する方法を示しています。

Console

バックアップ保持ポリシーを設定するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. AWS リージョンを変更するには、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. アクティブ状態のクラスターのクラスター ID をクリックして、そのクラスターのバックアップ保持ポリシーを管理します。
4. バックアップ保持ポリシーを変更するには、[アクション]、[バックアップ保持期間の変更] を選択します。

   [バックアップ保持期間の変更] ダイアログボックスが表示されます。
5. バックアップ保持期間 (日単位) に、7 日 ~ 379 日の値を入力します。

6. [バックアップ保持期間の変更] を選択する。

バックアップ保持ポリシーからバックアップを除外または含めるには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. バックアップを表示するには、ナビゲーションペインで [バックアップ] を選択します。
3. [準備完了] 状態のバックアップのバックアップ ID をクリックして、除外または含めます。
4. リポジトリの バックアップの詳細 ページで、次のいずれかのアクションを実行します。
   - 有効期限 に日付があるバックアップを除外するには、[アクション]、[有効期限を無効にする] の順に選択します。
   - 有効期限のないバックアップを含めるには、[アクション]、[クラスター保持ポリシーの使用] の順に選択します。

AWS CLI

バックアップ保持ポリシーを設定するには (AWS CLI)

- コマンドラインプロンプトで、modify-cluster コマンドを発行します。クラスター ID とバックアップ保持ポリシーを指定します。

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> \
                                --backup-retention-policy
 Type=DAYS,Value=<number of days to retain backups>
{
   "Cluster": {
      "BackupPolicy": "DEFAULT",
      "BackupRetentionPolicy": {
         "Type": "DAYS",
         "Value": 90
      },
      "Certificates": {},
      "ClusterId": "cluster-kdmrayrc7gi",
      "CreateTimestamp": 1504903546.035,
      "Hsms": [],
      "HsmType": "hsm1.medium",
      "SecurityGroup": "sg-40399d28",
```

```
      "State": "ACTIVE",
      "SubnetMapping": {
         "us-east-2a": "subnet-f1d6e798",
         "us-east-2c": "subnet-0e358c43",
         "us-east-2b": "subnet-40ed9d3b"
      },
      "TagList": [
         {
            "Key": "Cost Center",
            "Value": "12345"
         }
      ],
      "VpcId": "vpc-641d3c0d"
   }
}
```

バックアップ保持ポリシーからバックアップを除外するには (AWS CLI)

- コマンドラインプロンプトで、modify-backup-attributes コマンドを発行します。バックアップ ID を指定し、never-expires フラグを設定し、バックアップを保存します。

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
                                          --never-expires
{
   "Backup": {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "READY",
      "ClusterId": "cluster-dygnwhmscg5",
      "NeverExpires": true
   }
}
```

バックアップ保持ポリシーにバックアップを含めるには (AWS CLI)

- コマンドラインプロンプトで、modify-backup-attributes コマンドを発行します。バックアップ ID を指定し、バックアップ保持ポリシーにバックアップを含めるように no-never-expires フラグを設定します。これはサービスが最終的にバックアップを削除することを意味します。

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
```

```
                                                --no-never-expires
{
   "Backup": {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "READY",
      "ClusterId": "cluster-dygnwhmscg5",
      "NeverExpires": false
   }
}
```

AWS CloudHSM API

API を使用してバックアップの保存管理を管理する方法については、次のトピックを参照してください。

- ModifyCluster
- ModifyBackupAttributes

AWS リージョン間での AWS CloudHSM クラスターバックアップのコピー

AWS CloudHSM クラスターバックアップは、クロスリージョンレジリエンス、グローバルワークロード、ディザスタリカバリなど、さまざまな理由でリージョン間でコピーできます。バックアップをコピーすると、それらはコピー先のリージョンに CREATE_IN_PROGRESS のステータスを用いて現れます。正常にコピーが実行されると、バックアップのステータスは READY に変わります。コピーが失敗した場合、バックアップのステータスが DELETED に変わります。入力パラメータにエラーがないかどうかを確認し、オペレーションを再度実行する前に、指定した送信元バックアップのステータスが DELETED ではないことを確認します。バックアップ、あるいはバックアップからクラスターを作成する方法の詳細については、クラスターのバックアップ または バックアップからクラスターを作成する を参照してください。

次の点に注意してください。

- クラスターバックアップを送信先リージョンにコピーするには、適切な IAM ポリシーの許可がアカウントに必要です。バックアップを別のリージョンにコピーするには、バックアップがある送信元リージョンへのアクセスが IAM ポリシーで許可されている必要があります。リージョン間のコピーが完了したら、コピーしたバックアップを操作するために、IAM ポリシーを使用して、送信

先リージョンへのアクセスを許可する必要があります。これには CreateCluster オペレーションの使用が含まれます。詳細については、「IAM 管理者の作成」を参照してください。

- 元のクラスターと、送信先リージョンのバックアップから作成された可能性のあるクラスターはリンクされません。これらのクラスターは別々に管理する必要があります。詳細については、「クラスター」を参照してください。
- AWS パーティション間でバックアップをコピーすることはできません。バックアップは、 AWS 制限されたリージョンと同じパーティション内の標準リージョン間でコピーできます。

バックアップを異なるリージョン (コンソール) にコピーする

異なるリージョン (コンソール)にバックアップをコピーするには

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. AWS リージョンを変更するには、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、[バックアップ] を選択します。
4. バックアップを選択し、別のリージョンにコピーします。
5. 選択したバックアップをコピーするには、Actions, Copy backup to another region を選択します。

   [バックアップを別のリージョンにコピーする] ダイアログボックスが表示されます。
6. Destination region で、Select a region からリージョンを選択します。
7. (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。クラスターに複数のタグを追加するには、Add tag を選択します。
8. [Copy backup] (バックアップをコピー) を選択します。

バックアップを異なるリージョン (AWS CLI) にコピーする

バックアップ ID を判別するには、describe-backups コマンドを実行します。

異なるリージョン (AWS CLI) にバックアップをコピーする方法

- コマンドラインプロンプトで、 copy-backup-to-region コマンドを実行します。送信先のリージョンと、送信元バックアップのバックアップ ID を指定します。バックアップ ID を指定した場合は、関連付けられたバックアップがコピーされます。

```
$ aws cloudhsmv2 copy-backup-to-region --destination-region <destination region> \
```

```
                                        --backup-id <backup ID>
```

異なるリージョンにバックアップをコピーする (AWS CloudHSM API)

API を使用してバックアップを削除および復元する方法については、次のトピックを参照してください。

- CopyBackupToRegion

での共有バックアップの使用 AWS CloudHSM

CloudHSM は AWS Resource Access Manager (AWS RAM) と統合してリソース共有を有効にします。 AWS RAM は、一部の CloudHSM リソースを他の AWS アカウント または と共有できるようにするサービスです AWS Organizations。では AWS RAM、リソース共有を作成して、所有しているリソースを共有します。リソース共有は、共有するリソースと、それらを共有するコンシューマーを指定します。コンシューマーには以下が含まれます。

- の組織 AWS アカウント 内外に固有 AWS Organizations
- の組織内の組織単位 AWS Organizations
- の組織全体 AWS Organizations

詳細については AWS RAM、AWS RAM 「 ユーザーガイド」を参照してください。

このトピックでは、所有しているリソースの共有方法と、共有されているリソースの使用方法を説明します。

内容

- バックアップを共有するための前提条件
- バックアップの共有
- 共有バックアップの共有解除
- 共有バックアップの特定
- 共有バックアップのアクセス許可
- 請求と使用量測定

バックアップを共有するための前提条件

- バックアップを共有するには、 でバックアップを所有している必要があります AWS アカウント。つまり、自分のアカウントにそのリソースが割り当てられているか、プロビジョニングされている必要があります。自分と共有されているバックアップを共有することはできません。
- バックアップを共有するには、バックアップが READY 状態である必要があります。
- バックアップを組織または AWS Organizationsの組織単位と共有するには、 AWS Organizations との共有を有効にする必要があります。詳細については、「AWS RAM ユーザーガイド」の「Enable Sharing with AWS Organizations」を参照してください。

バックアップの共有

バックアップを他の と共有する場合 AWS アカウント、バックアップに保存されているキーとユーザーを含むクラスターをバックアップから復元できるようにします。

バックアップを共有するには、リソース共有に追加する必要があります。リソース共有とは、 AWS アカウント間で自身のリソースを共有するための AWS RAM リソースです。リソース共有では、共有対象のリソースと、共有先のコンシューマーを指定します。CloudHSM コンソールを使用してバックアップを共有すると、既存のリソース共有に追加されます。新しいリソース共有にバックアップを追加するには、まず AWS RAM コンソールを使用してリソース共有を作成する必要があります。

の組織に属 AWS Organizations していて、組織内での共有が有効になっている場合、組織内のコンシューマーには共有バックアップへのアクセス権が自動的に付与されます。これに該当しない場合、コンシューマーはリソースへの参加の招待を受け取り、その招待を受け入れた後で、共有バックアップに対するアクセス許可が付与されます。

AWS RAM コンソールまたは を使用して、所有しているバックアップを共有できます AWS CLI。

AWS RAM コンソールを使用して所有しているバックアップを共有するには

「AWS RAM ユーザーガイド」の「リソース共有の作成」を参照してください。

所有しているバックアップを共有するには (AWS RAM コマンド)

create-resource-share コマンドを使用します。

所有しているバックアップを共有するには (CloudHSM コマンド)

> **Important**
>
> CloudHSM PutResourcePolicy オペレーションを使用してバックアップを共有できますが、代わりに AWS Resource Access Manager (AWS RAM) を使用することをお勧めします。を使用すると AWS RAM 、ポリシーが作成され、一度に複数のリソースを共有でき、共有リソースの検出可能性が向上します。PutResourcePolicy を使用し、コンシューマーが共有したバックアップを記述できるようにする場合は、 AWS RAM PromoteResourceShareCreatedFromPolicy API オペレーションを使用してバックアップを標準の AWS RAM リソース共有に昇格させる必要があります。

put-resource-policy コマンドを使用します。

1. `policy.json` という名前のファイルを作成し、次のポリシーをコピーします。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal": {
      "AWS":"111122223333"
      },
      "Action":[
        "cloudhsm:CreateCluster",
        "cloudhsm:DescribeBackups"
       ],
       "Resource":"arn:aws:cloudhsm:us-west-2:111122223333:backup/backup-to-share"
    }
  ]
}
```

2. 共有したいバックアップ ARN と識別子で `policy.json` を更新します。次の例では、123456789012 で識別される AWS アカウントのルートユーザーに読み取り専用アクセスを許可します。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
     "Effect":"Allow",
     "Principal": {
        "AWS": [
          "123456789012"
      ]
    },
    "Action": [
      "cloudhsm:CreateCluster",
      "cloudhsm:DescribeBackups"
     ],
    "Resource":"arn:aws:cloudhsm:us-west-2:123456789012:backup/backup-123"
   }
 ]
}
```

> **Important**
>
> DescribeBackups にアクセス許可を付与できるのは、アカウントレベルのみです。バックアップを別の顧客と共有すると、そのアカウントで DescribeBackups アクセス許可を持つプリンシパルはバックアップの詳細を確認できます。

3. put-resource-policy コマンドを使用します。

```
$ aws cloudhsmv2 put-resource-policy --resource-arn <resource-arn> --policy file://
policy.json
```

> **Note**
>
> この時点で、共有先の顧客はバックアップを利用できますが、DescribeBackups コマンドを実行しても、共有されているという情報は表示されません。次のステップでは、

バックアップをレスポンスに含めるために AWS RAM リソース共有を昇格させる方法について説明します。

4. AWS RAM リソース共有 ARN を取得します。

```
$ aws ram list-resources --resource-owner SELF --resource-arns <backup-arn>
```

これにより、次のようなレスポンスが得られます。

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

レスポンスから、次の手順で使用する *<resource-share-arn>* 値をコピーします。

5. AWS RAM promote-resource-share-created-from-policy コマンドを実行します。

```
$ aws ram promote-resource-share-created-from-policy --resource-share-
arn <resource-share-arn>
```

6. リソース共有が昇格したことを検証するには、 AWS RAM get-resource-shares コマンドを実行します。

```
$ aws ram get-resource-shares --resource-owner SELF --resource-share-
arns <resource-share-arn>
```

ポリシーが昇格されると、レスポンスに表示される `featureSet` は `STANDARD` になります。これは、新しいアカウントが、バックアップの詳細を記述できることを意味します。

共有バックアップの共有解除

リソースの共有を解除すると、コンシューマーはリソースを使用してクラスターを復元できなくなります。コンシューマーは、共有バックアップから復元したクラスターに引き続きアクセスできます。

自己所有の共有バックアップの共有を解除するには、そのバックアップをリソース共有から削除する必要があります。これを行うには、 AWS RAM コンソールまたは を使用します AWS CLI。

AWS RAM コンソールを使用して所有している共有バックアップの共有を解除するには

「AWS RAM ユーザーガイド」の「リソース共有の更新」を参照してください。

所有している共有バックアップの共有を解除するには (AWS RAM コマンド)

disassociate-resource-share コマンドを使用します。

所有している共有バックアップの共有を解除するには (CloudHSM コマンド)

delete-resource-policy コマンドを使用します。

```
$ aws cloudhsmv2 delete-resource-policy --resource-arn <resource-arn>
```

共有バックアップの特定

コンシューマーは、CloudHSM コンソールと AWS CLIを使用して、共有されているバックアップを識別できます。

CloudHSM コンソールを使用して共有されているバックアップを特定するには

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM 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、[バックアップ] を選択します。
4. テーブルで、[共有バックアップ] タブを選択します。

を使用して共有されているバックアップを特定するには AWS CLI

describe-backups コマンドで `--shared` パラメータを使用して、共有されているバックアップを返します。

共有バックアップのアクセス許可

所有者のアクセス許可

バックアップ所有者は、共有バックアップを記述して管理し、それを使用してクラスターを復元できます。

コンシューマーのアクセス許可

バックアップコンシューマーは共有バックアップを変更することはできませんが、それを記述してクラスターを復元するために使用できます。

請求と使用量測定

バックアップの共有に追加料金はかかりません。

でクローンされたクラスター AWS CloudHSM

CloudHSM CLI を使用してリモートリージョンのクラスターを同期します。これは、そのリージョンのクラスターが最初に別のリージョンのクラスターのバックアップから作成された場合です。たとえば、クラスターを別のリージョン (デスティネーション) にコピーし、後で元のクラスター (ソース) からの変更を同期するとします。このようなシナリオでは、キーレプリケートコマンドとユーザーレプリケートコマンドを使用してクラスターを同期します。CloudHSM CLI をインストールしていない場合は、??? の手順を参照してください。

関連トピック

- user replicate
- key replicate
- リージョン間のバックアップのコピー

AWS CloudHSM リソースにタグを付ける

タグは、 AWS リソースに割り当てるラベルです。 AWS CloudHSM クラスターにタグを割り当てることができます。各タグは 1 つのタグキーと 1 つのタグ値で構成されており、いずれもお客様が定義します。たとえば、タグキーは Cost Center、タグ値は 12345 などです。タグキーは、クラスターごとに一意にする必要があります。

タグは、さまざまな目的で使用できます。1 つの一般的な用途は、 AWS コストの分類と追跡です。自社のカテゴリ、たとえばコストセンター、アプリケーション名、所有者を表すタグを適用すると、複数のサービスにわたってコストを分類することができます。 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。このレポートを使用して、すべての AWS CloudHSM コストを単一の明細項目として表示するのではなく、プロジェクトまたはアプリケーションの観点から AWS CloudHSM コストを表示できます。

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。

タグの追加、更新、一覧表示、削除を行うには、AWS CloudHSM コンソール、あるいは AWS SDK かコマンドラインツールのどちらかを使用できます。

トピック

- AWS CloudHSM リソースのタグを追加または更新する
- AWS CloudHSM リソースのタグを一覧表示する
- AWS CloudHSM リソースからタグを削除する

AWS CloudHSM リソースのタグを追加または更新する

タグの追加または更新は、AWS CloudHSM コンソール、AWS Command Line Interface (AWS CLI)、または AWS CloudHSM API で行うことができます。

タグを追加または更新するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. タグ付けするクラスターを選択します。
3. [タグ] を選択します。
4. タグを追加するには、次の操作を行います。

a. [タグの編集]、[タグの追加] の順に選択します。

b. [Key (キー)] にタグのキーを入力します。

c. (オプション) [Value (値)] にタグの値を入力します。

d. [保存] を選択します。

5. タグを更新するには、次の操作を行います。

a. [タグの編集] を選択します。

> **Note**
>
> 既存のタグのタグキーを更新すると、コンソールによって既存のタグが削除され、新しいタグが作成されます。

b. 新しいタグ値を入力します。

c. [保存] を選択します。

タグを追加または更新するには (AWS CLI)

1. コマンドプロンプトで、tag-resource コマンドを発行し、タグとタグ付けするクラスターの ID を指定します。クラスター ID がわからない場合は、describe-clusters コマンドを発行します。

```
$ aws cloudhsmv2 tag-resource --resource-id <cluster ID> \
                              --tag-list Key="<tag key>",Value="<tag value>"
```

2. タグを更新するには、同じコマンドを使用しますが、既存のタグキーを指定します。既存のタグに新しいタグ値を指定すると、タグは新しい値で上書きされます。

タグを追加または更新するには (AWS CloudHSM API)

- TagResource リクエストを送信します。タグとタグ付けするクラスターの ID を指定します。

AWS CloudHSM リソースのタグを一覧表示する

クラスターのタグは、AWS CloudHSM コンソール、、AWS CLIまたは AWS CloudHSM API から一覧表示できます。

タグを一覧表示するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. タグを一覧表示するクラスターを選択します。
3. [タグ] を選択します。

タグを一覧表示するには (AWS CLI)

- コマンドプロンプトで、list-tags コマンドを発行し、タグを一覧表示するクラスターの ID を指定します。クラスター ID がわからない場合は、describe-clusters コマンドを発行します。

```
$ aws cloudhsmv2 list-tags --resource-id <cluster ID>
{
    "TagList": [
        {
            "Key": "Cost Center",
            "Value": "12345"
        }
    ]
}
```

タグを一覧表示するには (AWS CloudHSM API)

- ListTags リクエストを送信し、タグを一覧表示するクラスターの ID を指定します。

AWS CloudHSM リソースからタグを削除する

AWS CloudHSM コンソール、、または AWS CloudHSM API を使用してAWS CLI、 AWS CloudHSM クラスターからタグを削除できます。

タグを削除するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. タグを削除するクラスターを選択します。
3. [タグ] を選択します。
4. [タグの編集] を選択し、削除するタグの [タグの削除] を選択します。
5. [保存] を選択します。

タグを削除するには (AWS CLI)

- コマンドプロンプトで untag-resource コマンドを発行し、削除するタグのタグキーと、タグを削除するクラスターの ID を指定します。を使用してタグ AWS CLI を削除する場合は、タグ値ではなくタグキーのみを指定します。

```
$ aws cloudhsmv2 untag-resource --resource-id <cluster ID> \
                                --tag-key-list "<tag key>"
```

タグを削除するには (AWS CloudHSM API)

- AWS CloudHSM API でUntagResourceリクエストを送信し、クラスターの ID と削除するタグを指定します。

AWS CloudHSM コマンドラインツール

AWS リソースの管理に使用する AWS Command Line Interface (AWS CLI) に加えて、 AWS CloudHSM には、HSMs。では AWS CloudHSM、使い慣れた CLI を使用してクラスターを管理し、CloudHSM コマンドラインツールを使用して HSM を管理します。

さまざまなコマンドラインツールが用意されています。

クラスターと HSM を管理するには

AWS CLIの CloudHSMv2 コマンドと AWSPowerShell モジュールの HSM2PowerShell コマンドレット

- これらのツールは、 AWS CloudHSM クラスターと HSMs。
- CLI で CloudHSMv2 コマンドのコマンドを使用するには、 をインストールして設定 AWS CLI する必要があります。
- AWSPowerShell module の HSM2 PowerShell コマンドレットは、Windows PowerShell モジュールとクロスプラットフォームの PowerShell Core モジュールで使用できます。

HSM ユーザーを管理するには

CloudHSM CLI

- CloudHSM CLI を使用して、ユーザーの作成、ユーザーの削除、ユーザーの一覧作成、ユーザーパスワードの変更、ユーザー多要素認証 (MFA) の更新を行います。AWS CloudHSM クライアントソフトウェアには含まれていません。このツールのインストールに関するガイダンスについては、「Install and configure CloudHSM CLI」を参照してください。

ヘルパーツール

2 つのツールを使用すると、 AWS CloudHSM ツールとソフトウェアライブラリを使用できます。

- configure ツールは、CloudHSM クライアント設定ファイルを更新します。これにより AWS CloudHSM 、 はクラスター内の HSMsを同期できます。

  AWS CloudHSM には 2 つのメジャーバージョンがあり、クライアント SDK 5 が最新です。クライアント SDK 3 (以前のシリーズ) よりも多くの、さまざまな利点があります。
- pkpspeed は、ソフトウェアライブラリに依存しない HSM ハードウェアのパフォーマンスを測定します。

以前の SDK 向けツール

キー管理ツール (KMU) を使用して、対称キーと非対称キーペアを作成、削除、インポート、エクスポートします。

- key_mgmt_util。このツールは、 AWS CloudHSM クライアントソフトウェアに含まれています。

CloudHSM 管理ツール (CMU) を使用して HSM ユーザーを作成および削除します (ユーザー管理タスクのクォーラム認証の実装を含む)。

- cloudhsm_mgmt_util。このツールは、 AWS CloudHSM クライアントソフトウェアに含まれています。

以下は、 AWS CloudHSMの管理と利用に使えるコマンドラインツールについて説明します。

トピック

- AWS CloudHSM ツールの設定
- AWS CloudHSM コマンドラインインターフェイス (CLI)
- AWS CloudHSM 管理ユーティリティ (CMU)
- AWS CloudHSM キー管理ユーティリティ (KMU)

AWS CloudHSM ツールの設定

AWS CloudHSM は、クラスター内のすべてのハードウェアセキュリティモジュール (HSM) 間でデータを自動的に同期します。configure ツールは、同期メカニズムが使用する設定ファイルの HSM データを更新します。configure を使用して、コマンドラインツールを使用する前に、特に、クラスター内の HSM が変更されたときに、HSM データを更新します。

AWS CloudHSM には、2 つの主要なクライアント SDK バージョンが含まれています。

- クライアント SDK 5: これは最新かつデフォルトの クライアント SDK です。クライアント SDK 5 による利点については、「AWS CloudHSM クライアント SDK 5 の利点」を参照してください。
- クライアント SDK 3: これは古いクライアント SDK です。プラットフォームおよび言語ベースのアプリケーションの互換性および管理ツール用のコンポーネントの完全なセットが含まれています。

クライアント SDK 3 から クライアント SDK 5 に移行する手順については、「AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行」を参照してください。

トピック

- AWS CloudHSM クライアント SDK 5 設定ツール
- AWS CloudHSM クライアント SDK 3 設定ツール

AWS CloudHSM クライアント SDK 5 設定ツール

AWS CloudHSM クライアント SDK 5 設定ツールを使用して、クライアント側の設定ファイルを更新します。

クライアント SDK 5 の各コンポーネントには、構成ツールのファイル名にコンポーネントの指定子を含む構成ツールが含まれています。たとえば、クライアント SDK 5 の PKCS #11 ライブラリには、Linux上 `configure-pkcs11` またはWindows上 `configure-pkcs11.exe` で名付けられた構成ツールが含まれています。

トピック

- AWS CloudHSM クライアント SDK 5 設定構文
- AWS CloudHSM クライアント SDK 5 設定パラメータ
- AWS CloudHSM クライアント SDK 5 の設定例
- ブートストラップ OpenSSL プロバイダー
- クライアント SDK 5 設定ツールの詳細設定
- AWS CloudHSM クライアント SDK 5 の関連トピック

AWS CloudHSM クライアント SDK 5 設定構文

次の表は、クライアント SDK 5 AWS CloudHSM の設定ファイルの構文を示しています。パラメータの詳細については、「 」を参照してくださいthe section called “パラメータ”

PKCS #11

```
Usage: configure-pkcs11[ .exe ] [OPTIONS]

Options:
      --disable-certificate-storage
          Disables Certificate Storage
```

```
      --enable-certificate-storage
          Enables Certificate Storage
  -a <HSM ENI IP>...
          The address of the HSM instance
      --cluster-id <CLUSTER ID>
          The id of the cluster containing the HSM instance(s)
      --disable-key-availability-check
          Disables key availability check during key use
      --enable-key-availability-check
          Enables key availability check during key use
      --disable-validate-key-at-init
          Disables parameter validation during initialization of crypto operations
      --enable-validate-key-at-init
          Enables parameter validation during initialization of crypto operations
      --endpoint <ENDPOINT>
          Specify the AWS CloudHSM API Endpoint
      --region <REGION>
          The region of the cluster
      --hsm-ca-cert <HSM CA CERTIFICATE FILE>
          The HSM CA certificate file
      --log-type <LOG TYPE>
          The log type [possible values: term, file]
      --log-file <LOG FILE>
          The log file
      --log-level <LOG LEVEL>
          The logging level [possible values: error, warn, info, debug, trace]
      --log-rotation <LOG ROTATION>
          The log rotation interval [possible values: never, hourly, daily]
      --default-retry-mode <RETRY MODE>
          The default method of retry to use for certain non-terminal failures
 [possible values: off, standard]
      --client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
          The client certificate used for TLS client-hsm mutual authentication
      --client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
          The client private key used for TLS client-hsm mutual authentication
  -h, --help
          Print help
```

OpenSSL

```
Usage: configure-dyn[ .exe ] [OPTIONS]

Options:
```

```
  -a <HSM ENI IP>...
          The address of the HSM instance
      --cluster-id <CLUSTER ID>
          The id of the cluster containing the HSM instance(s)
      --disable-key-availability-check
          Disables key availability check during key use
      --enable-key-availability-check
          Enables key availability check during key use
      --disable-validate-key-at-init
          Disables parameter validation during initialization of crypto operations
      --enable-validate-key-at-init
          Enables parameter validation during initialization of crypto operations
      --endpoint <ENDPOINT>
          Specify the AWS CloudHSM API Endpoint
      --region <REGION>
          The region of the cluster
      --hsm-ca-cert <HSM CA CERTIFICATE FILE>
          The HSM CA certificate file
      --log-type <LOG TYPE>
          The log type [possible values: term, file]
      --log-file <LOG FILE>
          The log file
      --log-level <LOG LEVEL>
          The logging level [possible values: error, warn, info, debug, trace]
      --log-rotation <LOG ROTATION>
          The log rotation interval [possible values: never, hourly, daily]
      --default-retry-mode <RETRY MODE>
          The default method of retry to use for certain non-terminal failures
 [possible values: off, standard]
      --client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
          The client certificate used for TLS client-hsm mutual authentication
      --client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
          The client private key used for TLS client-hsm mutual authentication
  -h, --help
          Print help
```

KSP

```
Usage: configure-ksp.exe [OPTIONS]

Options:
  -a <HSM ENI IP>...
          The address of the HSM instance
```

```
      --server-client-cert-file <CLIENT CERTIFICATE FILE>
          The client certificate used for TLS client-server mutual authentication
      --server-client-key-file <CLIENT KEY FILE>
          The client private key used for TLS client-server mutual authentication
      --cluster-id <CLUSTER ID>
          The id of the cluster containing the HSM instance(s)
      --disable-key-availability-check
          Disables key availability check during key use
      --enable-key-availability-check
          Enables key availability check during key use
      --disable-validate-key-at-init
          Disables parameter validation during initialization of crypto operations
      --enable-validate-key-at-init
          Enables parameter validation during initialization of crypto operations
      --endpoint <ENDPOINT>
          Specify the AWS CloudHSM API Endpoint
      --region <REGION>
          The region of the cluster
      --hsm-ca-cert <HSM CA CERTIFICATE FILE>
          The HSM CA certificate file
      --log-type <LOG TYPE>
          The log type [possible values: term, file]
      --log-file <LOG FILE>
          The log file
      --log-level <LOG LEVEL>
          The logging level [possible values: error, warn, info, debug, trace]
      --log-rotation <LOG ROTATION>
          The log rotation interval [possible values: never, hourly, daily]
      --default-retry-mode <RETRY MODE>
          The default method of retry to use for certain non-terminal failures
[possible values: off, standard]
      --client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
          The client certificate used for TLS client-hsm mutual authentication
      --client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
          The client private key used for TLS client-hsm mutual authentication
      --enable-sdk3-compatibility-mode
          Enables key file usage for KSP
      --disable-sdk3-compatibility-mode
          Disables key file usage for KSP
  -h, --help
          Print help
```

JCE

```
Usage: configure-jce[ .exe ] [OPTIONS]

Options:
  -a <HSM ENI IP>...
          The address of the HSM instance
      --cluster-id <CLUSTER ID>
          The id of the cluster containing the HSM instance(s)
      --disable-key-availability-check
          Disables key availability check during key use
      --enable-key-availability-check
          Enables key availability check during key use
      --disable-validate-key-at-init
          Disables parameter validation during initialization of crypto operations
      --enable-validate-key-at-init
          Enables parameter validation during initialization of crypto operations
      --endpoint <ENDPOINT>
          Specify the AWS CloudHSM API Endpoint
      --region <REGION>
          The region of the cluster
      --hsm-ca-cert <HSM CA CERTIFICATE FILE>
          The HSM CA certificate file
      --log-type <LOG TYPE>
          The log type [possible values: term, file]
      --log-file <LOG FILE>
          The log file
      --log-level <LOG LEVEL>
          The logging level [possible values: error, warn, info, debug, trace]
      --log-rotation <LOG ROTATION>
          The log rotation interval [possible values: never, hourly, daily]
      --default-retry-mode <RETRY MODE>
          The default method of retry to use for certain non-terminal failures
 [possible values: off, standard]
      --client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
          The client certificate used for TLS client-hsm mutual authentication
      --client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
          The client private key used for TLS client-hsm mutual authentication
  -h, --help
          Print help
```

CloudHSM CLI

```
Usage: configure-cli[ .exe ] [OPTIONS]

Options:
  -a <HSM ENI IP>...
          The address of the HSM instance
      --cluster-id <CLUSTER ID>
          The id of the cluster containing the HSM instance(s)
      --disable-key-availability-check
          Disables key availability check during key use
      --enable-key-availability-check
          Enables key availability check during key use
      --disable-validate-key-at-init
          Disables parameter validation during initialization of crypto operations
      --enable-validate-key-at-init
          Enables parameter validation during initialization of crypto operations
      --endpoint <ENDPOINT>
          Specify the AWS CloudHSM API Endpoint
      --region <REGION>
          The region of the cluster
      --hsm-ca-cert <HSM CA CERTIFICATE FILE>
          The HSM CA certificate file
      --log-type <LOG TYPE>
          The log type [possible values: term, file]
      --log-file <LOG FILE>
          The log file
      --log-level <LOG LEVEL>
          The logging level [possible values: error, warn, info, debug, trace]
      --log-rotation <LOG ROTATION>
          The log rotation interval [possible values: never, hourly, daily]
      --default-retry-mode <RETRY MODE>
          The default method of retry to use for certain non-terminal failures
 [possible values: off, standard]
      --client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
          The client certificate used for TLS client-hsm mutual authentication
      --client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
          The client private key used for TLS client-hsm mutual authentication
  -h, --help
          Print help
```

OpenSSL Provider

```
Usage: configure-openssl-provider[ .exe ] [OPTIONS]

Options:
  -a <HSM ENI IP>...
          The address of the HSM instance
      --cluster-id <CLUSTER ID>
          The id of the cluster containing the HSM instance(s)
      --disable-key-availability-check
          Disables key availability check during key use
      --enable-key-availability-check
          Enables key availability check during key use
      --disable-validate-key-at-init
          Disables parameter validation during initialization of crypto operations
      --enable-validate-key-at-init
          Enables parameter validation during initialization of crypto operations
      --endpoint <ENDPOINT>
          Specify the AWS CloudHSM API Endpoint
      --region <REGION>
          The region of the cluster
      --hsm-ca-cert <HSM CA CERTIFICATE FILE>
          The HSM CA certificate file
      --log-type <LOG TYPE>
          The log type [possible values: term, file]
      --log-file <LOG FILE>
          The log file
      --log-level <LOG LEVEL>
          The logging level [possible values: error, warn, info, debug, trace]
      --log-rotation <LOG ROTATION>
          The log rotation interval [possible values: never, hourly, daily]
      --default-retry-mode <RETRY MODE>
          The default method of retry to use for certain non-terminal failures
 [possible values: off, standard]
      --client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
          The client certificate used for TLS client-hsm mutual authentication
      --client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
          The client private key used for TLS client-hsm mutual authentication
  -h, --help
          Print help
```

AWS CloudHSM クライアント SDK 5 設定パラメータ

AWS CloudHSM クライアント SDK 5 を設定するためのパラメータのリストを次に示します。

-a ***<ENI IP address>***

指定した IP アドレスをクライアント SDK 5 設定ファイルに追加します。クラスター内の HSM の任意の ENI IP アドレスを入力します。このオプションの使用方法の詳細については、「クライアント SDK 5でブートストラップ」 を参照してください。

必須: はい

--hsm-ca-cert ***<customerCA certificate file path>***

EC2クライアントインスタンスをクラスターに接続するために使用する認証局（CA）証明書を格納するディレクトリへのパス。このファイルは、クラスターを初期化するときに作成します。デフォルトでは、システムはこのファイルを次の場所で検索します。

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Server

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

クラスターの初期化または証明書の配置の詳細については、「???」 および 「???」 を参照してください。

必須: いいえ

--cluster-id ***<cluster ID>***

`DescribeClusters` を呼び出して、クラスターIDに関連付けられたクラスターのすべての HSM Elastic Network Interface（ENI）IPアドレスを検索します。システムは ENI IP アドレスを設定 AWS CloudHSM ファイルに追加します。

> **Note**
>
> パブリックインターネットにアクセスできない VPC 内の EC2 インスタンスから `--cluster-id`パラメータを使用する場合は、接続するインターフェイス VPC エンドポイ

ントを作成する必要があります AWS CloudHSM。VPCエンドポイントの詳細については、??? を参照してください。

必須: いいえ

--endpoint ***<endpoint>***

`DescribeClusters` 呼び出しに使用する AWS CloudHSM API エンドポイントを指定します。このオプションは `--cluster-id` と組み合わせて設定する必要があります。

必須: いいえ

--region ***<region>***

クラスターのリージョンを指定します。このオプションは `--cluster-id` と組み合わせて設定する必要があります。

この `--region` パラメータを指定しない場合、システムは `AWS_DEFAULT_REGION` または `AWS_REGION` の環境変数の読み取りを試みてリージョンを選択します。これらの変数が設定されていない場合、環境変数で別のファイルを指定しない限り、AWS Config (通常は `~/.aws/config`) のプロファイルに関連付けられたリージョンをチェックします`AWS_CONFIG_FILE`。いずれも設定されていない場合は、`us-east-1` デフォルトでリージョンが設定されます。

必須: いいえ

--client-cert-hsm-tls-file ***<########## hsm tls ##>***

TLS クライアントと HSM の相互認証に使用するクライアント証明書へのパス。

このオプションは、CloudHSM CLI で HSM に少なくとも 1 つのトラストアンカーを登録している場合にのみ使用します。このオプションは `--client-key-hsm-tls-file` と組み合わせて設定する必要があります。

必須: いいえ

--client-key-hsm-tls-file ***<######## hsm tls ####***

TLS クライアントと HSM の相互認証に使用されるクライアントキーへのパス。

このオプションは、CloudHSM CLI で HSM に少なくとも 1 つのトラストアンカーを登録している場合にのみ使用します。このオプションは `--client-cert-hsm-tls-file` と組み合わせて設定する必要があります。

必須: いいえ

--log-level ***<error | warn | info | debug | trace>***

システムがログファイルに書き込むべき最小のログレベルを指定します。各レベルは前のレベルを含み、最小レベルはエラー、最大レベルはトレースとなります。つまり、エラーを指定すると、システムはログにエラーのみを書き込みます。トレースを指定すると、システムはエラー、警告、情報 (info)、およびデバッグメッセージをログに書き込みます。詳細については、「クライアント SDK 5 のログの記録」を参照してください。

必須: いいえ

--log-rotation ***<daily | weekly>***

システムがログをローテートする頻度を指定します。詳細については、「クライアント SDK 5 のログの記録」を参照してください。

必須: いいえ

--log-file ***<file name with path>***

システムがログファイルを書き込む場所を指定します。詳細については、「クライアント SDK 5 のログの記録」を参照してください。

必須: いいえ

--log-type ***<term | file>***

システムがログをファイルまたはターミナルのどちらに書き込むかを指定します。詳細については、「クライアント SDK 5 のログの記録」を参照してください。

必須: いいえ

-h | --help

ヘルプを表示します。

必須: いいえ

--disable-key-availability-check

キーの可用性クォーラムを無効にするためのフラグ。このフラグを使用して、 AWS CloudHSM がキー可用性クォーラムを無効にし、クラスター内の 1 つの HSM にのみ存在するキーを使用できることを示します。このフラグを使用してキーの可用性クォーラムを設定する方法については、「???」を参照してください。

必須: いいえ

--enable-key-availability-check

キーの可用性クォーラムを有効にするためのフラグ。このフラグを使用して、 AWS CloudHSM がキー可用性クォーラムを使用し、それらのキーがクラスター内の 2 つの HSMs に存在するまでキーを使用できないことを示します。このフラグを使用してキーの可用性クォーラムを設定する方法については、「???」を参照してください。

デフォルトでは有効になっています。

必須: いいえ

--disable-validate-key-at-init

このフラグを指定すると、その後の呼び出しでキーのパーミッションを確認するための初期化呼び出しをスキップできるため、パフォーマンスが向上します。注意して使用してください。

背景: PKCS #11 ライブラリの一部のメカニズムでは、初期化コールで後続のコールでキーを使用できるかどうかを検証するマルチパートオペレーションをサポートしています。これには HSM への検証呼び出しが必要で、オペレーション全体にレイテンシーが追加されます。このオプションを使用すると、後続の呼び出しを無効にし、パフォーマンスを向上させる可能性があります。

必須: いいえ

--enable-validate-key-at-init

初期化呼び出しを使用して、後続の呼び出しでキーに対する許可を検証するように指定します。これはデフォルトのオプションです。`enable-validate-key-at-init` を使用して、これらの初期化呼び出しを再開するには `disable-validate-key-at-init` を一時停止します。

必須: いいえ

AWS CloudHSM クライアント SDK 5 の設定例

これらの例は、 AWS CloudHSM クライアント SDK 5 の設定ツールを使用する方法を示しています。

クライアント SDK 5 のブートストラップ

Example

この例では、クライアント SDK 5 の HSM データを更新するための `-a` パラメータを使用しています。`-a` パラメータの場合は、クラスターのいずれかの HSM の IP アドレスが必要です。

PKCS #11 library

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP
 addresses>
```

OpenSSL Dynamic Engine

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

OpenSSL Dynamic Engine Provider

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider -a <HSM IP addresses>
```

Key Storage Provider (KSP)

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" -a <HSM IP
 addresses>
```

JCE provider

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP
 addresses>
```

CloudHSM CLI

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the
 HSMs>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI
 IPv4 / IPv6 addresses of the HSMs>
```

> **Note**
>
> `--cluster-id` パラメータは `-a <HSM_IP_ADDRESSES>` の代わりに使用できます。`--cluster-id` の使用要件については、「AWS CloudHSM クライアント SDK 5 設定ツール」を参照してください。

`-a` パラメータの詳細については、「the section called “パラメータ”」をご参照ください。

クライアント SDK 5 のクラスター、リージョン、エンドポイントの指定

Example

この例では、`cluster-id` パラメータを使用して、`DescribeClusters` 呼び出しを行うことにより、クライアント SDK 5 をブートストラップします。

PKCS #11 library

クライアント SDK 5 の Linux EC2 インスタンスを **`cluster-id`** 呼び出しでブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id <cluster-1234567>
```

クライアント SDK 5 用の Windows EC2 インスタンスを **`cluster-id`** でブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --cluster-
id <cluster-1234567>
```

OpenSSL Dynamic Engine

クライアント SDK 5 の Linux EC2 インスタンスを **cluster-id** 呼び出しでブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id <cluster-1234567>
```

Key Storage Provider (KSP)

クライアント SDK 5 用の Windows EC2 インスタンスを **cluster-id** でブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --cluster-
id <cluster-1234567>
```

JCE provider

クライアント SDK 5 の Linux EC2 インスタンスを **cluster-id** 呼び出しでブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id <cluster-1234567>
```

クライアント SDK 5 用の Windows EC2 インスタンスを **cluster-id** でブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --cluster-
id <cluster-1234567>
```

CloudHSM CLI

クライアント SDK 5 の Linux EC2 インスタンスを **cluster-id** 呼び出しでブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id <cluster-1234567>
```

クライアント SDK 5 用の Windows EC2 インスタンスを **cluster-id** でブートストラップするには

- クラスター ID `cluster-1234567` を使用して、クラスター内の HSMの IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-
id <cluster-1234567>
```

`--region` と `--endpoint` のパラメータと `cluster-id` のパラメータを組み合わせて、システムが `DescribeClusters` の呼び出しを行う方法を指定することができます。例えば、クラスターのリージョンが AWS CLI のデフォルトとして設定されているものと異なる場合、そのリージョンを使用するように `--region` パラメータを使用する必要があります。さらに、呼び出しに使用する AWS CloudHSM API エンドポイントを指定することもできます。これは、デフォルトの DNS ホスト名を使用しない VPC インターフェイスエンドポイントを使用するなど、さまざまなネットワーク設定に必要な場合があります AWS CloudHSM。

PKCS #11 library

カスタムエンドポイントとリージョンを使用して Linux EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id <cluster-1234567> --
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

カスタムエンドポイントとリージョンを使用して Windows EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --cluster-
id <cluster-1234567>--region <us-east-1> --endpoint <https://cloudhsmv2.us-
east-1.amazonaws.com>
```

OpenSSL Dynamic Engine

カスタムエンドポイントとリージョンを使用して Linux EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id <cluster-1234567> --
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Key Storage Provider (KSP)

カスタムエンドポイントとリージョンを使用して Windows EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --cluster-
id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-
east-1.amazonaws.com>
```

JCE provider

カスタムエンドポイントとリージョンを使用して Linux EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id <cluster-1234567> --
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

カスタムエンドポイントとリージョンを使用して Windows EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --cluster-
id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-
east-1.amazonaws.com>
```

CloudHSM CLI

カスタムエンドポイントとリージョンを使用して Linux EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id <cluster-1234567> --
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

カスタムエンドポイントとリージョンを使用して Windows EC2 インスタンスをブートストラップするには

- 設定ツールを使用して、カスタムリージョンとエンドポイントを持つクラスター内の HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-
id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-
east-1.amazonaws.com>
```

`--cluster-id`、`--region`、`--endpoint` パラメータの詳細については、the section called “パラメータ”を参照してください。

TLS クライアントと HSM の相互認証のためのクライアント証明書とキーの更新する

Example

この例では、 パラメータ`--client-cert-hsm-tls-file`と `--client-key-hsm-tls-file`パラメータを使用して、 のカスタムキーと SSL 証明書を指定して SSL を再設定する方法を示します。 AWS CloudHSM

PKCS #11 library

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
$ sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで ssl-client.pem、ssl-client.key を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して ssl-client.pem と ssl-client.key を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

OpenSSL Dynamic Engine

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで ssl-client.pem、ssl-client.key を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Key Storage Provider (KSP)

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して `ssl-client.pem` と `ssl-client.key` を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

JCE provider

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで `ssl-client.pem`、`ssl-client.key` を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して `ssl-client.pem` と `ssl-client.key` を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem> `
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

CloudHSM CLI

Linux のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. 構成ツールで `ssl-client.pem`、`ssl-client.key` を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
            --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Windows のクライアント SDK 5 で TLS クライアントと HSM の相互認証にカスタム証明書とキーを使用するには

1. キーと証明書を適切なディレクトリにコピーします。

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
```

```
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. PowerShell インタプリタでは、構成ツールを使用して ssl-client.pem と ssl-client.key を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem> `
            --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

--client-cert-hsm-tls-file、および --client-key-hsm-tls-file、パラメータの詳細については、the section called “パラメータ” を参照してください。

クライアントキーの耐久性設定を無効にする

Example

この例では --disable-key-availability-check パラメータを使用して、クライアントキーの耐久性設定を無効にします。単一の HSM でクラスターを実行するには、クライアントキーの耐久性設定を無効にする必要があります。

PKCS #11 library

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-availability-check
```

OpenSSL Dynamic Engine

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

OpenSSL Dynamic Engine Provider

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --disable-key-availability-
check
```

Key Storage Provider (KSP)

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-
key-availability-check
```

JCE provider

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-
key-availability-check
```

CloudHSM CLI

Linux でクライアント SDK 5 のクライアントキーの耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Windows でクライアント SDK 5 のクライアントキー耐久性を無効にするには

- 構成ツールを使用して、クライアントキーの耐久性設定を無効にします。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-
key-availability-check
```

`--disable-key-availability-check` パラメータの詳細については、「the section called “パラメータ”」をご参照ください。

ログ記録オプションの管理

Example

クライアント SDK 5 では、`log-file`、`log-level`、`log-rotation`、および `log-type` パラメータを使用して、ログを管理します。

> **Note**
>
> AWS Fargate や AWS Lambda などのサーバーレス環境用に SDK を設定するには、 AWS CloudHSM ログタイプを に設定することをお勧めしますterm。クライアントログは `stderr` に出力され、その環境に設定された CloudWatch Logs のロググループにキャプチャされます。

PKCS #11 library

デフォルトのログ記録の場所

- ファイルの場所を指定しない場合、システムはログを以下のデフォルトの場所に書き込みます。

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-pkcs11.log
  ```

  Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
  ```

ログ記録レベルを設定し、他のログ記録オプションはデフォルトのままにしておくには

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
  ```

ファイルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

ターミナルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
  ```

OpenSSL Dynamic Engine

デフォルトのログ記録の場所

- ファイルの場所を指定しない場合、システムはログを以下のデフォルトの場所に書き込みます。

  Linux
  ```

```
stderr
```

ログ記録レベルを設定し、他のログ記録オプションはデフォルトのままにしておくには

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
  ```

ファイルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

ターミナルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
  ```

OpenSSL Dynamic Engine Provider

デフォルトのログ記録の場所

- ファイルの場所を指定しない場合、システムはログを以下のデフォルトの場所に書き込みます。

  Linux

  ```
  stderr
  ```

ログ記録レベルを設定し、他のログ記録オプションはデフォルトのままにしておくには

- ```
  $ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-level info
  ```

ファイルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-type file --log-file <file name with path> --log-rotation daily --log-level info
  ```

ターミナルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-type term --log-level info
  ```

Key Storage Provider (KSP)

デフォルトのログ記録の場所

- ファイルの場所を指定しない場合、システムはログを以下のデフォルトの場所に書き込みます。

  Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-ksp.log
  ```

ログ記録レベルを設定し、他のログ記録オプションはデフォルトのままにしておくには

- ```
  PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-level info
  ```

ファイルのログ記録オプションを設定するには

- ```
  PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-type file --log-file <file name with path> --log-rotation daily --log-level info
  ```

ターミナルのログ記録オプションを設定するには

- ```
  PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-type
   term --log-level info
  ```

JCE provider

デフォルトのログ記録の場所

- ファイルの場所を指定しない場合、システムはログを以下のデフォルトの場所に書き込みます。

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```

ログ記録レベルを設定し、他のログ記録オプションはデフォルトのままにしておくには

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-level info
  ```

ファイルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

ターミナルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
  ```

CloudHSM CLI

デフォルトのログ記録の場所

- ファイルの場所を指定しない場合、システムはログを以下のデフォルトの場所に書き込みます。

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-cli.log
  ```

  Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
  ```

ログ記録レベルを設定し、他のログ記録オプションはデフォルトのままにしておくには

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-level info
  ```

ファイルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

ターミナルのログ記録オプションを設定するには

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
  ```

`log-file`、`log-level`、`log-rotation`、`log-type` のパラメータの詳細については、「the section called “パラメータ”」を参照してください。
  ```

クライアント SDK 5 の発行証明書を配置する

Example

この例では `--hsm-ca-cert` パラメータを使用して、クライアント SDK 5 の発行証明書の場所を更新します。

PKCS #11 library

Linux クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate
 file>
```

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --hsm-ca-
cert <customerCA certificate file>
```

OpenSSL Dynamic Engine

Linux クライアント SDK 5 の発行証明書を配置する

- 構成ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate
 file>
```

OpenSSL Dynamic Engine Provider

Linux クライアント SDK 5 の発行証明書を配置する

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --hsm-ca-cert <customerCA
 certificate file>
```

Key Storage Provider (KSP)

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --hsm-ca-
cert <customerCA certificate file>
```

JCE provider

Linux クライアント SDK 5 の発行証明書を配置する

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate
 file>
```

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --hsm-ca-
cert <customerCA certificate file>
```

CloudHSM CLI

Linux クライアント SDK 5 の発行証明書を配置する

- 設定ツールを使用して、発行証明書の場所を指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate
 file>
```

Windows クライアント SDK 5 の発行証明書を配置します。

- 設定ツールを使用して、発行証明書の場所を指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --hsm-ca-cert <customerCA certificate file>
```

`--hsm-ca-cert` パラメータの詳細については、「the section called “パラメータ”」をご参照ください。

ブートストラップ OpenSSL プロバイダー

configure-openssl-provider ツールを使用して OpenSSL プロバイダーのインストールをブートストラップし、 AWS CloudHSM クラスターに接続します。

OpenSSL プロバイダーをブートストラップするには

1. クラスター内の HSM の IP アドレスを使用して configure-openssl-provider コマンドを実行します。

   ```
   $ sudo /opt/cloudhsm/bin/configure-openssl-provider -a <HSM IP address>
   ```

   <HSM IP ####> をクラスター内の任意の HSM の IP アドレスに置き換えます。
2. OpenSSL プロバイダーがクラスターに接続できることを確認し、設定を確認します。

   ```
   $ openssl list -providers -provider-path /opt/cloudhsm/lib -provider cloudhsm
   ```

設定パラメータの詳細については、「」を参照してくださいthe section called “パラメータ”。

クライアント SDK 5 設定ツールの詳細設定

AWS CloudHSM クライアント SDK 5 設定ツールには、ほとんどのお客様が使用する一般的な機能の一部ではない高度な設定が含まれています。詳細設定には追加機能があります。

> **Important**
>
> 設定を変更した後、変更内容を反映させるためにアプリケーションを再起動する必要があります。

- PKCS #11 の詳細設定
  - 用の PKCS #11 ライブラリを使用した複数のスロット設定 AWS CloudHSM
  - の PKCS #11 ライブラリのコマンドを再試行する AWS CloudHSM
- OpenSSL の詳細設定
  - の OpenSSL の再試行コマンド AWS CloudHSM
- KSP の詳細設定
  - のキーストレージプロバイダー (KSP) の SDK3 互換性モード AWS CloudHSM
- JCE の詳細設定
  - JCE プロバイダーを使用した複数の AWS CloudHSM クラスターへの接続
  - の JCE の再試行コマンド AWS CloudHSM
  - の JCE を使用したキー抽出 AWS CloudHSM
- コマンドラインインターフェイス (CLI) AWS CloudHSM の詳細設定
  - CloudHSM CLI を使用した複数のクラスターへの接続

AWS CloudHSM クライアント SDK 5 の関連トピック

AWS CloudHSM クライアント SDK 5 の詳細については、以下の関連トピックを参照してください。

- DescribeClusters API operation
- describe-clusters AWS CLI
- Get-HSM2Cluster PowerShell cmdlet
- クライアント SDK 5 のブートストラップ
- OpenSSL プロバイダーのブートストラップ
- AWS CloudHSM VPC エンドポイント
- クライアント SDK 5 キーの耐久性設定の管理
- クライアント SDK 5 ログ記録
- mTLS の設定 (推奨)

AWS CloudHSM クライアント SDK 3 設定ツール

Client AWS CloudHSM SDK 3 設定ツールを使用してクライアントデーモンをブートストラップし、CloudHSM 管理ユーティリティ (CMU) を設定します。

トピック

- AWS CloudHSM クライアント SDK 3 設定構文
- AWS CloudHSM クライアント SDK 3 設定パラメータ
- AWS CloudHSM クライアント SDK 3 の設定例
- AWS CloudHSM クライアント SDK 3 設定に関連するトピック

AWS CloudHSM クライアント SDK 3 設定構文

次の表は、クライアント SDK 3 AWS CloudHSM の設定ファイルの構文を示しています。

```
configure -h | --help
          -a <ENI IP address>
          -m [-i <daemon_id>]
          --ssl --pkey <private key file> --cert <certificate file>
          --cmu <ENI IP address>
```

AWS CloudHSM クライアント SDK 3 設定パラメータ

AWS CloudHSM クライアント SDK 3 を設定するためのパラメータのリストを次に示します。

-h | --help

コマンド構文を表示します。

必須: はい

-a ***<ENI IP address>***

指定した HSM の Elastic Network Interface (ENI) IP アドレスを AWS CloudHSM 設定ファイルに追加します。クラスターのいずれかの HSM の ENI IP アドレスを入力します。どれを選択してもかまいません。

クラスター内の HSM の ENI IP アドレスを取得するには、DescribeClusters オペレーション、describe-clusters AWS CLI コマンド、または Get-HSM2Cluster PowerShell コマンドレットを使用します。

> **Note**
>
> -a configure コマンドを実行する前に、 AWS CloudHSM クライアントを停止します。その後、-aコマンドが完了したら、 AWS CloudHSM クライアントを再起動します。詳細については、例を参照してください。

このパラメータは、次の設定ファイルを編集します。

- /opt/cloudhsm/etc/cloudhsm_client.cfg: AWS CloudHSM クライアントと key_mgmt_util が使用します。
- /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg: 使用者 cloudhsm_mgmt_util。

AWS CloudHSM クライアントが起動すると、設定ファイルの ENI IP アドレスを使用してクラスターをクエリし、cluster.infoファイル (/opt/cloudhsm/daemon/1/cluster.info) をクラスター内のすべての HSMs の正しい ENI IP アドレスで更新します。

必須: はい

-m

CMU が使用する構成ファイルで HSM ENI IP アドレスを更新します。

> **Note**
>
> -m パラメータは、クライアント SDK 3.2.1 以前の CMU で使用するためのものです。クライアント SDK 3.3.0 以降の CMU については、CMU の HSM データ更新プロセスを簡略化する「--cmuパラメータ」を参照してください。

の -aパラメータを更新configureして AWS CloudHSM クライアントを起動すると、クライアントデーモンはクラスターをクエリし、クラスター内のすべての HSM の正しい HSMs IP アドレスでcluster.infoファイルを更新します。-mconfigure コマンドを実行すると、Cloudhsm_mgmt_util が使用する cluster.info 構成ファイルから cloudhsm_mgmt_util.cfg 構成ファイルに HSM IP アドレスがコピーされ、更新が完了します。

-a configure コマンドを実行する前に、必ず -m コマンドを実行し、 AWS CloudHSM クライアントを再起動してください。これにより、cluster.info から cloudhsm_mgmt_util.cfg にコピーされたデータが完全で正確であることを確認できます。

必須: はい

-i

代替クライアントデーモンを指定します。デフォルト値は AWS CloudHSM クライアントを表します。

デフォルト: 1

必須: いいえ

--ssl

クラスターの SSL キーおよび証明書を指定するプライベートキーおよび証明書に置き換えます。このパラメータを使用する場合には、`--pkey` および `--cert` パラメータが必要となります。

必須: いいえ

--pkey

新しいプライベートキーを指定します。プライベートキーが含まれているファイルのファイル名を入力します。

必須: はい (--ssl が指定されている場合) それ以外の場合は、使用しないでください。

--cert

新しい証明書を指定します。証明書が含まれているファイルのファイル名を入力します。証明書は、クラスターを初期化するために使用される自己署名証明書である `customerCA.crt` 証明書に連鎖する必要があります。詳細については、「クラスターの初期化」を参照してください。

必須: はい (--ssl が指定されている場合) それ以外の場合は、使用しないでください。

--cmu ***<ENI IP address>***

`-a` と `-m` のパラメータを 1 つのパラメータにまとめます。指定された HSM Elastic Network Interface (ENI) IP アドレス AWS CloudHSM を設定ファイルに追加し、CMU 設定ファイルを更新します。クラスター内の任意の HSM から IP アドレスを入力します。クライアント SDK 3.2.1 以前については、「クライアント SDK 3.2.1 以前での CMU の使用」 を参照してください。

必須: はい

AWS CloudHSM クライアント SDK 3 の設定例

これらの例は、 AWS CloudHSM クライアント SDK 3 の configure ツールを使用する方法を示しています。

Example: AWS CloudHSM クライアントと key_mgmt_util の HSM データを更新する

この例では、 の -aパラメータを使用してconfigure、 AWS CloudHSM クライアントと key_mgmt_util の HSM データを更新します。-a パラメータの場合は、クラスターのいずれかの HSM の IP アドレスが必要です。IP アドレスを取得するには、コンソールまたは AWS CLI を使用します。

HSM の IP アドレスを取得するには (コンソール)

1. https://console.aws.amazon.com/cloudhsm/home で AWS CloudHSM コンソールを開きます。
2. AWS リージョンを変更するには、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. クラスターの詳細ページを開くには、クラスターテーブルでクラスター ID を選択します。
4. IP アドレスを取得するには、HSM タブに移動します。IPv4 クラスターの場合は、ENI IPv4 アドレスにリストされているアドレスを選択します。デュアルスタッククラスターの場合は、ENI IPv4 アドレスまたは ENI IPv6 アドレスのいずれかを使用します。

HSM の IP アドレスを取得する (AWS CLI)

- describe-clusters から AWS CLIコマンドを実行して、HSM の IP アドレスを取得します。コマンドの出力では、HSM の IP アドレスは `EniIp` および `EniIpV6` の値です (デュアルスタッククラスターの場合)。

```
$ aws cloudhsmv2 describe-clusters
{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
                    "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733",
...
```

: HSM のデータを更新するには

1. -a パラメータを更新する前に、 AWS CloudHSM クライアントを停止します。これにより、configure がクライアントの設定ファイルを編集する間に発生する可能性がある競合を防ぎます。クライアントがすでに停止している場合は、このコマンドによる影響はないので、スクリプトで使用できます。

   Amazon Linux

   ```
   $ sudo stop cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client stop
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client stop
   ```

   CentOS 8

   ```
   $ sudo service cloudhsm-client stop
   ```

   RHEL 7

   ```
   $ sudo service cloudhsm-client stop
   ```

   RHEL 8

   ```
   $ sudo service cloudhsm-client stop
   ```

   Ubuntu 16.04 LTS

   ```
   $ sudo service cloudhsm-client stop
   ```

   Ubuntu 18.04 LTS

   ```
   $ sudo service cloudhsm-client stop
   ```

Windows

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合。

  AWS CloudHSM クライアントを起動したコマンドウィンドウで Ctrl+C を使用します。

2. このステップでは、configure -aconfigure の -a パラメータを使用して 10.0.0.9 ENI IP アドレスを設定ファイルに追加します。

Amazon Linux

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Amazon Linux 2

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

CentOS 7

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

CentOS 8

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

RHEL 7

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

RHEL 8

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Ubuntu 16.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Ubuntu 18.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a 10.0.0.9
```

3. 次に、 AWS CloudHSM クライアントを再起動します。起動した クライアントでは、設定ファイルの ENI IP アドレスを使用してクラスターにクエリを実行します。次に、クラスター内のすべての HSM の ENI IP アドレスを、`cluster.info` ファイルに書き込みます。

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 8

   ```
   $ sudo service cloudhsm-client start
   ```

   RHEL 7

   ```
   $ sudo service cloudhsm-client start
   ```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合。

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

コマンドが完了すると、 AWS CloudHSM クライアントと key_mgmt_util が使用する HSM データは完全で正確です。

Example: クライアント SDK 3.2.1 以前から CMU の HSM データを更新

この例では、-mconfigure コマンドを使用して、更新された HSM データを `cluster.info` ファイルから cloudhsm_mgmt_util が使用する `cloudhsm_mgmt_util.cfg` ファイルにコピーしています。クライアント SDK 3.2.1 以前と同梱されている CMU でこれを使用します。

- を実行する前に-m、 AWS CloudHSM クライアントを停止し、 -a コマンドを実行してから、前の例に示すように AWS CloudHSM クライアントを再起動します。これにより、`cluster.info` ファイルから `cloudhsm_mgmt_util.cfg` ファイルにコピーされたデータが完全で正確であることを確認できます。

Linux

```
$ sudo /opt/cloudhsm/bin/configure -m
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -m
```

Example: クライアント SDK 3.3.0 以降に同梱されている CMU の HSM データを更新

この例では、--cmu パラメータの configure コマンドにして CMU の HSM データを更新しています。クライアント SDK 3.3.0 以降と同梱されている CMU で使用します。CMU の使用方法の詳細については、「CloudHSM 管理ユーティリティ (CMU) を使用したユーザーの管理」そして「クライアント 3.2.1 SDK 以前CMUで を使用する」を参照してください。

- --cmu パラメータを使用して、クラスター内の HSM の IP アドレスを渡します。

  Linux

  ```
  $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
  ```

  Windows

  ```
  PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
  ```

AWS CloudHSM クライアント SDK 3 設定に関連するトピック

AWS CloudHSM クライアント SDK 3 の詳細については、以下の関連トピックを参照してください。

- AWS CloudHSM key_mgmt_util を設定する

AWS CloudHSM コマンドラインインターフェイス (CLI)

CloudHSM CLI は、管理者がユーザーを管理し、Crypto User が AWS CloudHSMのクラスター内のキーを管理するのに役立ちます。CLI には、ユーザーの作成、削除、一覧作成、パスワード変更、多

要素認証 (MFA) の更新を行うツールが含まれています。また、キーの生成、削除、インポート、エクスポート、属性の取得および設定、キーの検索、暗号化オペレーションの実行などを行う複数のコマンドが含まれています。

CloudHSM CLI ユーザーの定義済みリストについては、「CloudHSM CLI による HSM ユーザー管理」を参照してください。CloudHSM CLI のキー属性の定義済みリストについては、「CloudHSM CLI のキー属性」を参照してください CloudHSM CLI でキーを管理する方法については、「CloudHSM CLI によるキー管理」を参照してください。

クイックスタートについては、「コマンドラインインターフェイス (CLI) AWS CloudHSM の開始方法」を参照してください。CloudHSM CLI コマンドとコマンドの使用例の詳細については、「CloudHSM CLI コマンドのリファレンス」を参照してください。

トピック

- AWS CloudHSM コマンドラインインターフェイス (CLI) がサポートするプラットフォーム
- コマンドラインインターフェイス (CLI) AWS CloudHSM の開始方法
- CloudHSM CLI のコマンドモード
- CloudHSM CLI のキー属性
- CloudHSM CLI の詳細設定
- CloudHSM CLI コマンドのリファレンス

AWS CloudHSM コマンドラインインターフェイス (CLI) がサポートするプラットフォーム

このトピックでは、 CLI がサポートする Linux および Windows AWS CloudHSM プラットフォームについて説明します。

Linux サポート

サポートされているプラットフォーム	x86_64 アーキテクチャ	ARM アーキテクチャ
Amazon Linux 2	はい	はい
Amazon Linux 2023	はい	はい

サポートされているプラットフォーム	x86_64 アーキテクチャ	ARM アーキテクチャ
Red Hat Enterprise Linux 8 (8.3 以降)	はい	はい
Red Hat Enterprise Linux 9 (9.2 以降)	はい	はい
Red Hat Enterprise Linux 10 (10.0 以降)	はい	はい
Ubuntu 22.04 LTS	はい	はい
Ubuntu 24.04 LTS	はい	はい
Ubuntu 26.04 LTS	はい	はい

- SDK 5.16 は、Ubuntu 20.04 LTS プラットフォームをサポートする最後のリリースでした。詳細については、「Ubuntu のウェブサイト」を参照してください。
- SDK 5.12 は、CentOS 7 (7.8 以降) プラットフォームをサポートする最後のリリースでした。詳細については、「CentOS のウェブサイト」を参照してください。
- SDK 5.12 は、Red Hat Enterprise Linux 7 (7.8 以降) プラットフォームをサポートする最後のリリースでした。詳細については、Red Hat のウェブサイトを参照してください。
- SDK 5.4.2 は、CentOS 8 プラットフォームをサポートする最後のリリースでした。詳細については、「CentOS のウェブサイト」を参照してください。

Windows サポート

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019
- Microsoft Windows Server 2022
- Microsoft Windows Server 2025

コマンドラインインターフェイス (CLI) AWS CloudHSM の開始方法

CloudHSM CLI コマンドラインインターフェイス (CLI) を使用すると、 AWS CloudHSM クラスター内のユーザーを管理できます。このトピックを使用して、ユーザーの作成、ユーザーのリスト、CloudHSM CLI のクラスターへの接続など、基本的なハードウェアセキュリティモジュール (HSM) ユーザー管理タスクを開始します。

トピック

- CloudHSM CLI をインストールする
- CloudHSM CLI を使用する

CloudHSM CLI をインストールする

CloudHSM CLI をダウンロードしてインストールするには、次のコマンドを使用します AWS CloudHSM。

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

ARM64 アーキテクチャの Amazon Linux 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-
latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

ARM64 の Amazon Linux 2 アーキテクチャ:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

x86_64 アーキテクチャの RHEL 10:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.x86_64.rpm
```

ARM64 アーキテクチャの RHEL 10:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

ARM64 アーキテクチャの RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

ARM64 アーキテクチャの RHEL 8:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.aarch64.rpm
```

Ubuntu 26.04 LTS

x86_64 アーキテクチャの Ubuntu 26.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/cloudhsm-cli_latest_u26.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u26.04_amd64.deb
```

ARM64 アーキテクチャの Ubuntu 26.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/cloudhsm-cli_latest_u26.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u26.04_arm64.deb
```

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-cli_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_amd64.deb
```

ARM64 アーキテクチャの Ubuntu 24.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-cli_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

ARM64 アーキテクチャの Ubuntu 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 の場合、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 の場合、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 の場合、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

次のコマンドを使用して CloudHSM CLI を設定します。

クライアント SDK 5 の Linux EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the
 HSMs>
```

クライアント SDK 5 の Windows EC2 インスタンスをブートストラップするには

- 構成ツールを使用して、クラスターの HSM の IP アドレスを指定します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI
 IPv4 / IPv6 addresses of the HSMs>
```

CloudHSM CLI を使用する

CloudHSM CLI を起動して使用するには、次のコマンドを使用します。

1. CloudHSM CLI を起動するには、次のコマンドを使用します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
   ```

2. login コマンドを使用して、クラスターにログインします。このコマンドはすべてのユーザーが使用できます。

   次のコマンド例では管理者でログインしています。これは、デフォルトの管理者アカウントです。このユーザーのパスワードは、「クラスターのアクティブ化」を行う場合に設定します。

   ```
   aws-cloudhsm > login --username admin --role admin
   ```

   システムからパスワードの入力を求められます。パスワードを入力するとコマンドが正常に実行されたことが出力で示されます。

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

3. user list コマンドを実行して、クラスター上のすべてのユーザーを一覧表示します。

   ```
   aws-cloudhsm > user list
   {
     "error_code": 0,
     "data": {
       "users": [
   ```

```
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

4. user create を使用して、**example_user** という名前の CU ユーザーを作成します。

   前のステップで管理者ユーザーとしてログインしたので、CU を作成できます。ユーザーの作成および削除や、他のユーザーのパスワード変更などのユーザー管理作業を行うことができるのは、管理者ユーザーのみです。

```
aws-cloudhsm > user create --username example_user --role crypto-user
Enter password:
Confirm password:
{
 "error_code": 0,
 "data": {
   "username": "example_user",
   "role": "crypto-user"
 }
}
```

5. user list を使用して、クラスター上のすべてのユーザーを一覧表示します。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
```

```
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "example_user",
        "role": "crypto_user",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

6. logout コマンドを使用して AWS CloudHSM クラスターからログアウトします。

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. quit コマンドを使用して CLI を停止します。

```
aws-cloudhsm > quit
```

CloudHSM CLI のコマンドモード

CloudHSM CLI では、シングルコマンドモードおよびインタラクティブモードの 2 つの方法でコマンドを実行できます。インタラクティブモードはユーザー向けに設計され、シングルコマンドモードはスクリプト向けに設計されています。

> **Note**
>
> すべてのコマンドはインタラクティブモードおよびシングルコマンドモードで動作します。

インタラクティブモード

次のコマンドを使用して CloudHSM CLI インタラクティブモードを起動

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

CLI をインタラクティブモードで使用する場合、login コマンドを使用してユーザーアカウントにログインできます。

```
aws-cloudhsm > login --username <USERNAME> --role ROLE>
```

CloudHSM CLI コマンドをすべて一覧表示するには、次のコマンドを実行します。

```
aws-cloudhsm > help
```

CloudHSM CLI コマンドの構文を取得するには、次のコマンドを実行します。

```
aws-cloudhsm >  help <command-name>
```

HSM 上のユーザーのリストを取得するには、「user list」と入力します。

```
aws-cloudhsm > user list
```

CloudHSM CLI のセッションを終了するには、次のコマンドを実行します。

```
aws-cloudhsm > quit
```

シングルコマンドモード

> **Note**
>
> 単一コマンドモードを使用する場合は、シェルによって解釈される可能性のある環境変数とコマンドライン引数の特殊文字をエスケープする必要があります。

シングルコマンドモードを使用して CloudHSM CLI を実行する場合、認証情報を提供するために CLOUDHSM_PIN と CLOUDHSM_ROLE という 2 つの環境変数を設定する必要があります。

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```

これを実行すると、環境に保存されている認証情報を使用してコマンドを実行できます。

```
$ cloudhsm-cli user change-password --username alice --role crypto-user
Enter password:
Confirm password:
{
    "error_code": 0,
    "data": {
      "username": "alice",
      "role": "crypto-user"
    }
}
```

CloudHSM CLI のキー属性

このトピックでは、CloudHSM CLI を使用してキー属性を設定する方法について説明します。CloudHSM CLI のキー属性では、キーのタイプ、キーの機能、またはキーのラベル付け方法を定義できます。一部の属性は固有の特性 (キーのタイプなど) を定義します。その他の属性は「true」ま

たは「false」に設定できます。これらの属性を変更すると、キーの機能の一部が有効または無効になります。

キー属性の使用方法を示す例については、親コマンド CloudHSM CLI のキーカテゴリ の下にリストされているコマンドを参照してください。

以下のトピックでは、CloudHSM CLI のキー属性に関するその他の詳細について説明します。

トピック

- CloudHSM CLI のサポートされている属性
- CloudHSM CLI の値を確認する
- CloudHSM CLI の関連トピック

CloudHSM CLI のサポートされている属性

ベストプラクティスとして、制限する属性の値のみを設定してください。値を指定しない場合、CloudHSM CLI は次の表で指定されたデフォルト値を使用します。

次の表は、CloudHSM CLI のキー属性、指定できる値、デフォルト、および関連する注意事項の一覧です。Value 列のセルが空の場合は、属性に割り当てられている特定のデフォルト値がないことを示します。

CloudHSM CLI 属性	値	key set-attributeで変更可能	キー作成時に設定可能
always-sensitive	sensitive が常に True に設定されており、変更されたことがない場合、値は True となります。	いいえ	いいえ
check-value	キーのチェック値。詳細については、「その他の詳細」を参照してください。	いいえ	いいえ
class	想定される値: secret-ke	いいえ	はい

CloudHSM CLI 属性	値	key set-attributeで変更可能	キー作成時に設定可能
	y 、public-key 、private-key 。		
curve	EC キーペア生成に使用される楕円曲線。 有効な値: secp224r1 、secp256r1 、、prime256v1 、secp384r1 secp256k1 、secp521r1 、および ed25519 ed25519 は、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。	いいえ	EC では設定可能、RSA では設定不可
decrypt	デフォルト: False	はい	はい
derive	デフォルト: False	hsm2m.medium インスタンスでは、導出を設定できます。hsm1.medium インスタンスでは、RSA キーに対して導出を設定することはできません。	はい
destroyable	デフォルト: True	はい	はい

CloudHSM CLI 属性	値	key set-attributeで変更可能	キー作成時に設定可能
ec-point	EC キーの場合、ANSI X9.62 ECPoint 値「Q」の DER エンコーディングを 16 進数形式で表します。 他のキータイプの場合、この属性は存在しません。	いいえ	いいえ
encrypt	デフォルト: False	はい	はい
extractable	デフォルト: True	hsm2m.medium インスタンスでは true から false に変更できますが、false から true に変更することはできません。hsm1.medium インスタンスでは変更できません。	はい
id	デフォルト: 空	hsm2m.medium インスタンスでは、id を設定できます。hsm1.medium インスタンスでは、id を設定することはできません。	はい

CloudHSM CLI 属性	値	key set-attributeで変更可能	キー作成時に設定可能
key-length-bytes	AES キーを生成するために必要です。 有効な値: 16 バイト、24 バイト、32 バイト。	いいえ	いいえ
key-type	指定できる値: aes、rsa、ec、および ml-dsa	いいえ	はい
label	デフォルト: 空	はい	はい
local	デフォルト: HSM で生成されたキー向け True、HSM にインポートされたキー向け False。	いいえ	いいえ
modifiable	デフォルト: True	true から false に変更できますが、false から true には変更できません。	はい
mldsa-algorithm	ML-DSA キーペアの生成に使用される ML-DSA アルゴリズム。 有効な値: ML-DSA-44、ML-DSA-65、ML-DSA-87	いいえ	はい

CloudHSM CLI 属性	値	key set-attributeで変更可能	キー作成時に設定可能
modulus	RSA キーペアを生成するために使用されたモジュラス。他のキータイプの場合、この属性は存在しません。	いいえ	いいえ
modulus-size-bits	RSA キーペアを生成するために必要です。 最小値は 2048 です。	いいえ	RSA では設定可能、EC では設定不可
never-extractable	この値は、抽出可能が False に設定されたことがない場合、True となります。 この値は、抽出可能が True に設定されたことがある場合、False となります。	いいえ	いいえ
private	デフォルト: True	いいえ	はい
public-exponent	RSA キーペアを生成するために必要です。 有効な値: 値は、65537 以上の奇数にする必要があります	いいえ	RSA では設定可能、EC では設定不可

CloudHSM CLI 属性	値	key set-attributeで変更可能	キー作成時に設定可能
sensitive	デフォルト: • この値は AES キー、EC および RSA プライベートキーでは True となります。 • この値は EC および RSA パブリックキー用では False となります。	いいえ	プライベートキーでは設定可能で、パブリックキーでは設定できません。
sign	デフォルト: • 値は AES キーでは True となります。 • 値は RSA キーと EC キーでは False となります。	はい	はい
token	デフォルト: True	false から true に変更できますが、true から false には変更できません。	はい
trusted	デフォルト: False	このパラメータを設定できるのは管理者ユーザーのみです。	いいえ
unwrap	デフォルト: False	はい	はい。ただし、パブリックキーは除きます。

CloudHSM CLI 属性	値	key set-attributeで変更可能	キー作成時に設定可能
unwrap-template	値は、このラッピングキーを使用してラップ解除されたキーに適用される属性テンプレートを使用する必要があります。	はい	いいえ
verify	デフォルト: • 値は AES キーでは True となります。 • 値は RSA キーと EC キーでは False となります。	はい	はい
wrap	デフォルト: False	はい	はい。ただし、プライベートキーは除きます。
wrap-template	値は、属性テンプレートを使用し、このラッピングキーでラップされたキーと一致させる必要があります。	はい	いいえ
wrap-with-trusted	デフォルト: False	はい	はい

CloudHSM CLI の値を確認する

CloudHSM CLI のチェック値は、HSM がキーをインポートまたは生成するときに生成されるキーの 3 バイトのハッシュまたはチェックサムです。キーをエクスポートした後など、HSM の外部でチェック値を計算することもできます。次に、チェック値を比較して、キーのアイデンティティと整

合性を確認できます。キーのチェック値を取得するには、キーリストに Verbose (詳細) フラグを付けて使用します。

AWS CloudHSM は、次の標準メソッドを使用してチェック値を生成します。

- 対称キー: ゼロブロックをキーで暗号化した結果の最初の 3 バイト。
- 非対称キーペア: 公開キーの SHA-1 ハッシュの最初の 3 バイト。
- HMAC キー : 現時点では、HMAC キーの KCV はサポートされていません。

CloudHSM CLI の関連トピック

CloudHSM CLI の詳細については、次のトピックを参照してください。

- CloudHSM CLI のキーカテゴリ
- CloudHSM CLI コマンドのリファレンス

CloudHSM CLI の詳細設定

コマンドラインインターフェイス (CLI) AWS CloudHSM には、次の高度な設定が含まれています。これは、ほとんどのお客様が使用する一般的な設定の一部ではありません。これらの設定には追加機能があります。

- 複数のクラスターへの接続

CloudHSM CLI を使用した複数のクラスターへの接続

AWS CloudHSM クライアント SDK 5 では、単一の CLI インスタンスから複数の CloudHSM クラスターへの接続を許可するように CloudHSM CLI を設定できます。

以下のトピックでは、CloudHSM CLI マルチクラスター機能を使用して複数のクラスターに接続する方法について説明します。

トピック

- のマルチクラスターの前提条件 AWS CloudHSM
- マルチクラスター機能用に CloudHSM CLI を設定する
- AWS CloudHSM 設定にクラスターを追加する

- AWS CloudHSM 設定からクラスターを削除する
- で複数のクラスターを操作する AWS CloudHSM

のマルチクラスターの前提条件 AWS CloudHSM

複数のクラスターに接続する AWS CloudHSM ように でクラスターを設定する前に、次の前提条件を満たす必要があります。

- 接続先の 2 つ以上の AWS CloudHSM クラスターとそのクラスター証明書。
- セキュリティグループが上記のすべてのクラスターに接続するように正しく設定された EC2 インスタンス。クラスターとクライアントインスタンスのセットアップ方法の詳細については、「 の開始方法 AWS CloudHSM」を参照してください。
- マルチクラスター機能を設定するには、CloudHSM CLI を事前にダウンロードしてインストールしておく必要があります。これをまだ確認していない場合は、「???」の手順を参照してください。
- `./configure-cli[.exe] -a` で設定されたクラスターは `cluster-id` に関連付けられないため、アクセスできません。このガイドで説明されているように、`config-cli add-cluster` に従うことで再設定できます。

マルチクラスター機能用に CloudHSM CLI を設定する

CloudHSM CLI をマルチクラスター機能用に設定するには、次の手順に従います。

1. 接続するクラスターを特定します。
2. 以下に示すように configure-cli のサブコマンド `add-cluster` を使用して、これらのクラスターを CloudHSM CLI 設定に追加します。
3. 新しい設定を有効にするには、実行されている CloudHSM CLI プロセスがあれば再起動します。

AWS CloudHSM 設定にクラスターを追加する

複数のクラスターに接続する場合は、`configure-cli add-cluster` コマンドを使用してクラスターを設定に追加します。

構文

```
configure-cli add-cluster [OPTIONS]
```

```
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
        [--client-cert-hsm-tls-file <CLIENT CERTIFICATE FILE>]
        [--client-key-hsm-tls-file <CLIENT KEY FILE>]
        [-h, --help]
```

例

cluster-id パラメータを使用してクラスターを追加する

Example

configure-cli add-cluster とともに cluster-id パラメータを使用して、クラスター (cluster-1234567 の ID) を設定に追加します。

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" add-cluster --
cluster-id <cluster-1234567>
```

> **Tip**
>
> configure-cli add-cluster を cluster-id パラメータと一緒に使用してもクラスターが追加されない場合は、以下の例を参照して、追加するクラスターを識別するための --region と --endpoint パラメータも必要な、より長いバージョンのこのコマンドを参照してください。例えば、クラスターのリージョンが AWS CLI のデフォルトとして設定されているものと異なる場合、適切なリージョンを使用するように --region パラメータを使用する必要があります。さらに、呼び出しに使用する AWS CloudHSM API エンドポイントを指定することもできます。これは、デフォルトの DNS ホスト名を使用しない VPC インターフェイスエンドポイントを使用するなど、さまざまなネットワーク設定に必要な場合があります AWS CloudHSM。

cluster-id、endpoint、および region パラメータを使用してクラスターを追加する

Example

configure-cli add-cluster とともに cluster-id、endpoint、region のパラメータを使用して、クラスター (cluster-1234567 の ID) を設定に追加します。

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" add-cluster --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

--cluster-id、--region、--endpoint パラメータの詳細については、the section called “パラメータ”を参照してください。

パラメータ

--cluster-id *<Cluster ID>*

DescribeClusters を呼び出して、クラスターIDに関連付けられたクラスターのすべての HSM Elastic Network Interface (ENI) IPアドレスを検索します。システムは ENI IP アドレスを設定 AWS CloudHSM ファイルに追加します。

> **Note**
>
> パブリックインターネットにアクセスできない VPC 内の EC2 インスタンスから --cluster-idパラメータを使用する場合は、接続するインターフェイス VPC エンドポイントを作成する必要があります AWS CloudHSM。VPC エンドポイントの詳細については、「???」を参照してください。

必須: はい

--endpoint ***<Endpoint>***

`DescribeClusters` 呼び出しに使用する AWS CloudHSM API エンドポイントを指定します。このオプションは `--cluster-id` と組み合わせて設定する必要があります。

必須: いいえ

--hsm-ca-cert ***<HsmCA Certificate Filepath>***

HSM CA 証明書ファイルへのファイルパスを指定します。

必須: いいえ

--region ***<Region>***

クラスターのリージョンを指定します。このオプションは `--cluster-id` と組み合わせて設定する必要があります。

この `--region` パラメータを指定しない場合、システムは `AWS_DEFAULT_REGION` または `AWS_REGION` の環境変数の読み取りを試みてリージョンを選択します。これらの変数が設定されていない場合、環境変数で別のファイルを指定しない限り、AWS Config (通常は `~/.aws/config`) のプロファイルに関連付けられたリージョンをチェックします`AWS_CONFIG_FILE`。いずれも設定されていない場合は、`us-east-1` デフォルトでリージョンが設定されます。

必須: いいえ

--client-cert-hsm-tls-file ***<########## hsm tls ##>***

TLS クライアントと HSM の相互認証に使用するクライアント証明書へのパス。

このオプションは、CloudHSM CLI で HSM に少なくとも 1 つのトラストアンカーを登録している場合にのみ使用します。このオプションは `--client-key-hsm-tls-file` と組み合わせて設定する必要があります。

必須: いいえ

--client-key-hsm-tls-file ***<######### hsm tls ####***

TLS クライアントと HSM の相互認証に使用されるクライアントキーへのパス。

このオプションは、CloudHSM CLI で HSM に少なくとも 1 つのトラストアンカーを登録している場合にのみ使用します。このオプションは `--client-cert-hsm-tls-file` と組み合わせて設定する必要があります。

必須: いいえ

AWS CloudHSM 設定からクラスターを削除する

CloudHSM CLI で複数のクラスターに接続する場合は、`configure-cli remove-cluster` コマンドを使用して設定からクラスターを削除します。

構文

```
configure-cli remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

例

`cluster-id` パラメータを使用してクラスターを削除します

Example

`configure-cli remove-cluster` とともに `cluster-id` パラメータを使用して、クラスター (`cluster-1234567` の ID) を設定から削除します。

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli remove-cluster --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" remove-cluster --
cluster-id <cluster-1234567>
```

`--cluster-id` パラメータの詳細については、「the section called “パラメータ”」をご参照ください。

パラメータ

--cluster-id ***<Cluster ID>***

設定から削除するクラスターの ID。

必須: はい

で複数のクラスターを操作する AWS CloudHSM

CloudHSM CLI で複数のクラスターを設定したら、cloudhsm-cli コマンドを使用してクラスターとやり取りします。

例

インタラクティブモードを使用する場合のデフォルト **cluster-id** の設定

Example

cluster-id パラメータとともに ??? を使用して、デフォルトのクラスター (cluster-1234567 という ID) を設定します。

Linux

```
$ cloudhsm-cli interactive --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive --
cluster-id <cluster-1234567>
```

単一のコマンドを実行するときに **cluster-id** を設定する

Example

cluster-id パラメータを使用して、??? の取得元のクラスター (cluster-1234567 という ID) を設定します。

Linux

```
$ cloudhsm-cli cluster hsm-info --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" cluster hsm-info
 --cluster-id <cluster-1234567>
```

CloudHSM CLI コマンドのリファレンス

CloudHSM CLI は、管理者が AWS CloudHSM クラスター内のユーザーを管理するのに役立ちます。CloudHSM CLI は、インタラクティブモードとシングルコマンドモードの 2 つのモードで実行できます。クイックスタートについては、「コマンドラインインターフェイス (CLI) AWS CloudHSM の開始方法」を参照してください。

多くの CloudHSM CLI コマンドを実行するには、CloudHSM CLI を起動し、HSM にログインする必要があります。HSM を追加または削除する場合は、CloudHSM CLI の構成ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

以下のトピックでは、CloudHSM CLI のコマンドについて説明します。

コマンド	説明	ユーザータイプ
アクティブ化	CloudHSM クラスターをアクティブ化し、クラスターが新しいものであることを確認します。これは他のオペレーションを実行する前に行う必要があります。	非アクティブ管理者
hsm-info	クラスター内の HSM を一覧表示します。	認証されていないユーザーを含むすべての[1]。ログインは必須ではありません。
ecdsa	EC プライベートキーと ECDSA 署名機構を使用して署名を生成します。	Crypto User (CU)
ed25519	Ed25519 プライベートキーと PureEdDSA 署名メカニズムを使用して署名を生成します。	CU
ed25519ph	Ed25519 プライベートキーと HashEdDSA 署名メカニズムを使用して署名を生成します。	CU

コマンド	説明	ユーザータイプ
ml-dsa	ML-DSA[2] プライベートキーと ML-DSA 署名メカニズムを使用して署名を生成します。	CU
rsa-pkcs	RSA プライベートキーと RSA-PKCS 署名機構を使用して署名を生成します。	CU
rsa-pkcs-pss	RSA プライベートキーと RSA-PKCS-PSS 署名機構を使用して署名を生成します。	CU
ecdsa	ファイルが特定のパブリックキーによって HSM で署名されていることを確認します。ECDSA 署名機構を使用して署名が生成されたことを確認します。署名されたファイルをソース ファイルと比較し、指定された ecdsa パブリックキーと署名機構に基づいて暗号的に関連するかどうかを分析します。	CU
ed25519	Ed25519 パブリックキーを使用して PureEdDSA 署名を検証します。	CU
ed25519ph	Ed25519 パブリックキーを使用して HashEdDSA 署名を検証します。	CU
ml-dsa	ML-DSA[2] パブリックキーと ML-DSA 検証メカニズムを使用して署名を検証します。	CU

コマンド	説明	ユーザータイプ
rsa-pkcs	ファイルが特定のパブリックキーによって HSM で署名されていることを確認します。RSA-PKCS 署名機構を使用して署名が生成されたことを確認します。署名されたファイルをソース ファイルと比較し、指定された rsa パブリックキーと署名機構に基づいて暗号的に関連するかどうかを分析します。	CU
rsa-pkcs-pss	ファイルが特定のパブリックキーによって HSM で署名されていることを確認します。RSA-PKCS-PSS 署名機構を使用して署名が生成されたことを確認します。署名されたファイルをソース ファイルと比較し、指定された rsa パブリックキーと署名機構に基づいて暗号的に関連するかどうかを分析します。	CU
キー削除	AWS CloudHSM クラスターからキーを削除します。	CU
key generate-file	AWS CloudHSM クラスターにキーファイルを生成します。	CU
key generate-asymmetric-pair rsa	AWS CloudHSM クラスターに非対称 RSA キーペアを生成します。	CU

コマンド	説明	ユーザータイプ
key generate-asymmetric-pair ec	AWS CloudHSM クラスターに非対称楕円曲線 (EC) キーペアを生成します。	CU
キーgenerate-asymmetric-pair ml-dsa	AWS CloudHSM クラスターに非対称 ML-DSA キーペアを生成します。	CU
key generate-symmetric aes	AWS CloudHSM クラスターに対称 AES キーを生成します。	CU
key generate-symmetric generic-secret	AWS CloudHSM クラスターに対称汎用シークレットキーを生成します。	CU
key import pem	PEM 形式キーを HSM にインポートします。このコマンドを使用すると、HSM の外部で生成されたパブリックキーをインポートできます。	CU
キーリスト	AWS CloudHSM クラスター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。	CU
key replicate	ソースクラスターから複製先のクラスターにキーをレプリケートします。	CU
key set-attribute	AWS CloudHSM クラスター内のキーの属性を設定します。	CU はこのコマンドを実行でき、管理者は信頼できる属性を設定できます。
キーシェア	AWS CloudHSM クラスター内の他の CUs とキーを共有します。	CU

コマンド	説明	ユーザータイプ
キー共有解除	AWS CloudHSM クラスター内の他の CUs とキーの共有を解除します。	CU
aes-gcm	AES ラッピングキーと AES-GCM ラップ解除メカニズムを使用して、ペイロードキーをクラスターにラップ解除します。	CU
aes-no-pad	AES ラッピングキーと AES-NO-PAD ラップ解除メカニズムを使用して、ペイロードキーをクラスターにラップ解除します。	CU
aes-pkcs5-pad	AES ラッピングキーと AES-PKCS5-PAD ラップ解除メカニズムを使用してペイロードキーをラップ解除します。	CU
aes-zero-pad	AES ラッピングキーと AES-ZERO-PAD ラップ解除メカニズムを使用して、ペイロードキーをクラスターにラップ解除します。	CU
cloudhsm-aes-gcm	AES ラッピングキーと CLOUDHSM-AES-GCM ラップ解除メカニズムを使用して、ペイロードキーをクラスターにラップ解除します。	CU

コマンド	説明	ユーザータイプ
rsa-aes	RSA プライベートキーと RSA-AES ラップ解除メカニズムを使用してペイロードキーをラップ解除します。	CU
rsa-oaep	RSA プライベートキーと RSA-OAEP ラップ解除メカニズムを使用してペイロードキーをラップ解除します。	CU
rsa-pkcs	RSA プライベートキーと RSA-PKCS ラップ解除メカニズムを使用してペイロードキーをラップ解除します。	CU
aes-gcm	HSM の AES キーと AES-GCM ラップメカニズムを使用してペイロードキーをラップします。	CU
aes-no-pad	HSM の AES キーと AES-NO-PAD ラップメカニズムを使用してペイロードキーをラップします。	CU
aes-pkcs5-pad	HSM の AES キーと AES-PKCS5-PAD ラップメカニズムを使用してペイロードキーをラップします。	CU
aes-zero-pad	HSM の AES キーと AES-ZERO-PAD ラップメカニズムを使用してペイロードキーをラップします。	CU

コマンド	説明	ユーザータイプ
cloudhsm-aes-gcm	HSM の AES キーと CLOUDHSM-AES-GCM ラッピングメカニズムを使用してペイロードキーをラップします。	CU
rsa-aes	HSM の RSA パブリックキーと RSA-AES ラップメカニズムを使用してペイロードキーをラップします。	CU
rsa-oaep	HSM の RSA パブリックキーと RSA-OAEP ラップメカニズムを使用してペイロードキーをラップします。	CU

コマンド	説明	ユーザータイプ
CloudHSM CLI の key wrap rsa-pkcs コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の RSA パブリックキーと RSA-PKCS ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの `extractable` 属性を `true` に設定する必要があります。 キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。 key wrap rsa-pkcs コマンドを使用するには、まず AWS CloudHSM クラスターに RSA キーが必要です。CloudHSM CLI の generate-asymmetric-pair カテゴリ コマンドと `true` に設定された `wrap` 属性を使用して、RSA キーペアを生成できます。 ユーザーのタイプ このコマンドは、次のタイプのユーザーが実行できます。 • Crypto User (CU) 要件 • このコマンドを実行するには、CU としてログインする必要があります。	HSM の RSA パブリックキーと RSA-PKCS ラップメカニズムを使用してペイロードキーをラップします。	CU

コマンド	説明	ユーザータイプ
login (ログイン)	AWS CloudHSM クラスターにログインします。	Admin、Crypto User (CU)、および Appliance User (AU)
logout (サインアウト)	AWS CloudHSM クラスターからログアウトします。	Admin、CU、および Appliance User (AU)
quorum token-sign delete	クォーラム承認サービスのトークンを 1 つ以上削除します。	管理者
quorum token-sign generate	クォーラム承認サービスのトークンを生成します。	管理者
quorum token-sign list	CloudHSM クラスターに存在するすべてのトークン署名のクォーラムトークンを一覧表示します。	認証されていないユーザーを含むすべての [1]。ログインは必須ではありません。
quorum token-sign list-quorum-values	CloudHSM クラスターに設定されているクォーラム値を一覧表示します。	認証されていないユーザーを含むすべての [1]。ログインは必須ではありません。
quorum token-sign set-quorum-value	クォーラム認定サービスの新しいクォーラム値を設定します。	管理者
user change-mfa	ユーザーの多要素認証 (MFA) 戦略を変更します。	Admin、CU
user change-password	HSM 上のユーザーのパスワードを変更します。どのユーザーも自分のパスワードを変更できます。管理者は誰でもパスワードを変更できます。	Admin、CU
user create	AWS CloudHSM クラスターにユーザーを作成します。	管理者

コマンド	説明	ユーザータイプ
user delete	AWS CloudHSM クラスター内のユーザーを削除します。	管理者
user list	AWS CloudHSM クラスター内のユーザーを一覧表示します。	認証されていないユーザーを含むすべての [1]。ログインは必須ではありません。
ユーザー変更クォーラムトークン署名登録	ユーザーのクォーラムトークン署名クォーラム戦略を登録します。	管理者

注釈

- [1] すべてのユーザーには、リストされているすべてのロールとログインしていないユーザーが含まれます。
- [2] 2026 年 9 月 1 日以降、ML-DSA は hsm2m.medium クラスターの FIPS モードで使用できます。

CloudHSM CLI のクラスターカテゴリ

CloudHSM CLI では、cluster はコマンドグループの親カテゴリであり、親カテゴリと組み合わせるとクラスター固有のコマンドを作成します。現在、クラスターカテゴリは以下のコマンドで構成されています。

トピック

- CloudHSM CLI でクラスターをアクティブ化する
- CloudHSM CLI で HSM を一覧表示する
- CloudHSM CLI のクラスター mtls カテゴリ

CloudHSM CLI でクラスターをアクティブ化する

CloudHSM CLI の cluster activate コマンドを使用して AWS CloudHSMで新しいクラスターをアクティブ化します。クラスターを使用して暗号化オペレーションを実行するには、まずこのコマンドを実行する必要があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- 非アクティブ管理者

構文

このコマンドにはパラメータはありません。

```
aws-cloudhsm > help cluster activate
Activate a cluster

This command will set the initial Admin password. This process will cause your CloudHSM
 cluster to
move into the ACTIVE state.

USAGE:
    cloudhsm-cli cluster activate [OPTIONS] [--password <PASSWORD>]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --password <PASSWORD>
          Optional: Plaintext activation password If you do not include this argument
 you will be prompted for it

  -h, --help
          Print help (see a summary with '-h')
```

例

このコマンドは、管理者ユーザーの初期パスワードを設定してクラスターをアクティブ化します。

```
aws-cloudhsm > cluster activate
Enter password:
Confirm password:
{
  "error_code": 0,
```

```
  "data": "Cluster activation successful"
}
```

関連トピック

- user create
- user delete
- user change-password

CloudHSM CLI で HSM を一覧表示する

CloudHSM CLI の cluster hsm-info コマンドを使用して、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSMsを一覧表示します。このコマンドは、CloudHSM CLI にログインしていなくても実行できます。

> **Note**
>
> HSMs を追加または削除する場合は、 AWS CloudHSM クライアントとコマンドラインツールが使用する設定ファイルを更新します。そうしないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- すべてのユーザー。このコマンドは、ログインしていなくても実行できます。

Syntax

```
aws-cloudhsm > help cluster hsm-info
List info about each HSM in the cluster

Usage: cloudhsm-cli cluster hsm-info [OPTIONS]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
```

```
  -h, --help                      Print help
```

例

このコマンドは、 AWS CloudHSM クラスターに存在する HSMsを一覧表示します。

```
aws-cloudhsm > cluster hsm-info
{
  "error_code": 0,
  "data": {
    "hsms": [
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000590",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000625",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000663",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
```

```
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      }
    ]
  }
}
```

出力には以下の属性があります。

- Vendor: HSM のベンダー名。
- Model: HSM のモデル番号。
- Serial-number: デバイスのシリアル番号。置換により変更される場合があります。
- Hardware-version-major: ハードウェアのメジャーバージョン。
- Hardware-version-minor: ハードウェアのマイナーバージョン。
- Firmware-version-major: メジャーファームウェアバージョン。
- Firmware-version-minor: マイナーファームウェアバージョン。
- Firmware-build-number: ファームウェアビルド番号。
- Firmware-id: ファームウェア ID。ビルドに加えてメジャーバージョンとマイナーバージョンが含まれます。
- FIPS-state: クラスターとその中の HSM の FIPS モード。FIPS モードの場合、出力は「2 [単一要素認証の FIPS モード]」です。非 FIPS モードの場合、出力は「0 [単一要素認証の非 FIPS モード]」です。

関連トピック

- CloudHSM CLI でクラスターをアクティブ化する

CloudHSM CLI のクラスター mtls カテゴリ

CloudHSM CLI では、 cluster mtlsはコマンドグループの親カテゴリであり、親カテゴリと組み合わせると、 AWS CloudHSM クラスターに固有のコマンドが作成されます。現在、このカテゴリは次のコマンドで構成されています。

トピック

- CloudHSM CLI でトラストアンカーの登録を解除する
- CloudHSM CLI で mTLS 適用レベルを取得する
- CloudHSM CLI でトラストアンカーを一覧表示する
- CloudHSM CLI でトラストアンカーを登録する
- CloudHSM CLI で mTLS 適用レベルを設定する

CloudHSM CLI でトラストアンカーの登録を解除する

CloudHSM CLI の cluster mtls deregister-trust-anchor コマンドを使用して、クライアントと AWS CloudHSMの間の相互 TLS のトラストアンカーを登録解除します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者

要件

- このコマンドを実行するには、管理者ユーザーとしてログインする必要があります。

Syntax

```
aws-cloudhsm > help cluster mtls deregister-trust-anchor

Deregister a trust anchor for mtls

Usage: cluster mtls deregister-trust-anchor [OPTIONS] --certificate-reference
 [<CERTIFICATE_REFERENCE>...]

Options:
      --certificate-reference <CERTIFICATE_REFERENCE>  A hexadecimal or decimal
 certificate reference
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --approval <APPROVAL>  Filepath of signed quorum token file to approve operation
  -h, --help                     Print help
```

例

Example

次の例では、このコマンドは HSM からトラストアンカーを削除します。

```
aws-cloudhsm > cluster mtls deregister-trust-anchor --certificate-reference 0x01

{
  "error_code": 0,
  "data": {
    "message": "Trust anchor with reference 0x01 deregistered successfully"
  }
}
```

その後、次のように list-trust-anchors コマンドを実行して、トラストアンカーが AWS CloudHSMから登録解除されたことを確認できます。

```
aws-cloudhsm > cluster mtls list-trust-anchors

{
  "error_code": 0,
  "data": {
    "trust_anchors": []
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<CERTIFICATE_REFERENCE>

16 進数または 10 進数の証明書リファレンス。

必須: はい

> **Warning**
>
> クラスター内のトラストアンカーの登録を解除すると、そのトラストアンカーによって署名されたクライアント証明書を使用した既存の mTLS 接続はすべて削除されます。

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。クォーラムクラスターサービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- cluster mtls reregister-trust-anchor
- cluster mtls list-trust-anchors
- mTLS の設定 (推奨)

CloudHSM CLI で mTLS 適用レベルを取得する

CloudHSM CLI の cluster mtls get-enforcement コマンドを使用して、クライアントと AWS CloudHSMの間の相互 TLS の使用の適用レベルを取得します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者
- Crypto User (CU)

要件

- このコマンドを実行するには、管理者ユーザーまたは Crypto User (CU) としてログインする必要があります。

Syntax

```
aws-cloudhsm > help cluster mtls get-enforcement
```

```
Get the status of mtls enforcement in the cluster

Usage: cluster mtls get-enforcement [OPTIONS]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

例

Example

次の例では、このコマンドは AWS CloudHSMの mtls 適用レベルを一覧表示します。

```
aws-cloudhsm > cluster mtls get-enforcement

{
  "error_code": 0,
  "data": {
    "mtls-enforcement-level": "none"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

関連トピック

- cluster mtls set-enforcement
- mTLS の設定 (推奨)

CloudHSM CLI でトラストアンカーを一覧表示する

CloudHSM CLI の cluster mtls list-trust-anchors コマンドを使用して、クライアントと AWS CloudHSMの間の相互 TLS に使用できるすべてのトラストアンカーを一覧表示します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- すべてのユーザー。このコマンドは、ログインしていなくても実行できます。

Syntax

```
aws-cloudhsm > help cluster mtls list-trust-anchors

List all trust anchors for mtls

Usage: cluster mtls list-trust-anchors [OPTIONS]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

例

Example

次の例では、このコマンドは AWS CloudHSMから登録されたすべてのトラストアンカーを一覧表示します。

```
aws-cloudhsm > cluster mtls list-trust-anchors

{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate 1>",
```

```
        "cluster-coverage": "full"
      },
      {
        "certificate-reference": "0x02",
        "certificate": "<PEM Encoded Certificate 2>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

関連トピック

- cluster mtls reregister-trust-anchor
- cluster mtls deregister-trust-anchor
- mTLS の設定 (推奨)

CloudHSM CLI でトラストアンカーを登録する

CloudHSM CLI の cluster mtls register-trust-anchor コマンドを使用して、クライアントと AWS CloudHSMの間の相互 TLS のトラストアンカーを登録します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者

要件

は、次のキータイプのトラストアンカー AWS CloudHSM を受け入れます。

キータイプ	説明
EC	secp256r1 (P-256)、secp384r1 (P-384)、および secp521r1 (P-521) 曲線。
RSA	2048 ビット、3072 ビット、および 4096 ビットの RSA キー。

Syntax

```
aws-cloudhsm > help cluster mtls register-trust-anchor

Register a trust anchor for mtls

Usage: cluster mtls register-trust-anchor [OPTIONS] --path [<PATH>...]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --path <PATH>  Filepath of the trust anchor to register
      --approval <APPROVAL>  Filepath of signed quorum token file to approve operation
  -h, --help                     Print help
```

例

Example

次の例では、このコマンドはトラストアンカーを HSM に登録します。登録できるトラストアンカーの最大数は 2 個です。

```
aws-cloudhsm > cluster mtls register-trust-anchor --path /home/rootCA

{
  "error_code": 0,
  "data": {
    "trust_anchor": {
      "certificate-reference": "0x01",
      "certificate": "<PEM Encoded Certificate>",
      "cluster-coverage": "full"
```

```
    }
  }
}
```

その後、次のように list-trust-anchors コマンドを実行して、トラストアンカーが AWS CloudHSMに登録されたことを確認できます。

```
aws-cloudhsm > cluster mtls list-trust-anchors

{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PATH>

登録するトラストアンカーのファイルパス。

必須: はい

> **Note**
>
> AWS CloudHSM では、中間証明書をトラストアンカーとして登録できます。このような場合は、PEM でエンコードされた証明書チェーンファイル全体を、証明書を階層順にして、HSM に登録する必要があります。

AWS CloudHSM は、6980 バイトの証明書チェーンをサポートします。

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。クォーラムクラスターサービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- cluster mtls deregister-trust-anchor
- cluster mtls list-trust-anchors
- mTLS の設定 (推奨)

CloudHSM CLI で mTLS 適用レベルを設定する

CloudHSM CLI の cluster mtls set-enforcement コマンドを使用して、クライアントと AWS CloudHSMの間の相互 TLS の使用の適用レベルを設定します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- ユーザー名 admin を使用する管理者

要件

このコマンドを実行するには:

- 少なくとも 1 つのトラストアンカーが AWS CloudHSMに正常に登録されました。
- 適切なプライベートキーとクライアント証明書を使用して CloudHSM CLI を設定し、相互 TLS 接続で CloudHSM CLI を起動します。
- デフォルトの管理者として「admin」というユーザー名でログインする必要があります。他の管理者ユーザーは、このコマンドを実行できません。

Syntax

```
aws-cloudhsm > help cluster mtls set-enforcement
```

```
Set mtls enforcement policy in the cluster

Usage: cluster mtls set-enforcement [OPTIONS] --level [<LEVEL>...]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --level <LEVEL>  Level to be set for mtls in the cluster [possible values: none,
 cluster]
      --approval <APPROVAL>  Filepath of signed quorum token file to approve operation
  -h, --help                     Print help
```

例

Example

次の例では、このコマンドは の mtls 適用レベル AWS CloudHSM をクラスターに設定します。set-enforcement コマンドは、相互 TLS 接続で、かつ、ユーザー名を admin とする管理者ユーザーとしてログインした場合のみ実行できます。「AWS CloudHSMに対して mTLS 適用を設定する」を参照してください。

```
aws-cloudhsm > cluster mtls set-enforcement --level cluster

{
  "error_code": 0,
  "data": {
    "message": "Mtls enforcement level set to Cluster successfully"
  }
}
```

その後、get-enforcement コマンドを実行して、適用レベルがクラスターに設定されたことを確認できます。

```
aws-cloudhsm > cluster mtls get-enforcement

{
  "error_code": 0,
  "data": {
```

```
    "mtls-enforcement-level": "cluster"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<LEVEL>

クラスター内の mtls に設定するレベル。

有効な値

- cluster: クライアントとクラスター AWS CloudHSM 間の相互 TLS の使用を強制します。
- none: クライアントとクラスター AWS CloudHSM 間の相互 TLS の使用を強制しないでください。

必須: はい

> **Warning**
>
> クラスターで mTLS の使用を適用すると、既存の非 mTLS 接続はすべて削除され、mTLS 証明書を持つクラスターにのみ接続できます。

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。クォーラムクラスターサービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- cluster mtls get-enforcement
- mTLS の設定 (推奨)

CloudHSM CLI の暗号化カテゴリ

CloudHSM CLI では、crypto はコマンドグループの親カテゴリであり、親カテゴリと組み合わせると暗号オペレーション固有のコマンドを作成します。現在、このカテゴリは次のコマンドで構成されています。

- サイン
  - ecdsa
  - ed25519
  - ed25519ph
  - ml-dsa
  - rsa-pkcs
  - rsa-pkcs-pss
- 検証
  - ecdsa
  - ed25519
  - ed25519ph
  - ml-dsa
  - rsa-pkcs
  - rsa-pkcs-pss

CloudHSM CLI の暗号化署名カテゴリ

CloudHSM CLI では、crypto sign はコマンドグループの親カテゴリであり、親カテゴリと組み合わせると、 AWS CloudHSM クラスター内で選択したプライベートキーを使用して署名を生成します。crypto sign には、次のサブコマンドがあります。

- CloudHSM CLI で ECDSA メカニズムを使用して署名を生成する
- CloudHSM CLI で PureEdDSA メカニズムを使用して署名を生成する
- CloudHSM CLI で HashEdDSA メカニズムを使用して署名を生成する
- CloudHSM CLI で ML-DSA メカニズムを使用して署名を生成する
- CloudHSM CLI で RSA-PKCS メカニズムを使用して署名を生成する
- CloudHSM CLI で RSA-PKCS-PSS メカニズムを使用して署名を生成する

crypto sign を使用するには、HSM 内にプライベートキーが必要です。プライベートキーは、次のコマンドで生成できます。

- key generate-asymmetric-pair ec
- キーgenerate-asymmetric-pair ml-dsa
- key generate-asymmetric-pair rsa

CloudHSM CLI で ECDSA メカニズムを使用して署名を生成する

CloudHSM CLI の crypto sign ecdsa コマンドを使用して、EC プライベートキーと ECDSA 署名メカニズムを使用して署名を生成します。

crypto sign ecdsa コマンドを使用するには、まず AWS CloudHSM クラスターに EC プライベートキーが必要です。`sign` 属性を `true` に設定して、CloudHSM CLI を使用して非対称 EC キーペアを生成する コマンドを使用して EC プライベートキーを生成できます。

生成される ECDSA 署名は、r と のコンポーネントが raw バイナリデータとして連結され`r||s`、base64 でエンコードされた形式で返される 形式で生成されます。

> **Note**
>
> 署名は、 CloudHSM CLI の暗号化検証カテゴリ サブコマンド AWS CloudHSM を使用して で検証できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help crypto sign ecdsa
```

```
Sign with the ECDSA mechanism

Usage: crypto sign ecdsa --key-filter [<KEY_FILTER>>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be signed
      --data <DATA>
          Base64 Encoded data to be signed
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
      --data-type <DATA_TYPE>
          The type of data passed in, either raw or digest [possible values: raw,
 digest]
  -h, --help
          Print help
```

例

これらの例は、crypto sign ecdsa を使用して、ECDSA 署名メカニズムと SHA256 ハッシュ関数を使用して署名を生成する方法を示しています。このコマンドは HSM でプライベートキーを使用します。

Example例: ベース 64 でエンコードされたデータの署名を生成する

```
aws-cloudhsm > crypto sign ecdsa \
    --key-filter attr.label=ec-private \
    --hash-function sha256 \
    --data YWJjMTIz
{
  "error_code": 0,
  "data": {
```

```
    "key-reference": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

Example例: データファイルの署名を生成する

```
aws-cloudhsm > crypto sign ecdsa \
    --key-filter attr.label=ec-private \
    --hash-function sha256 \
    --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データパスを通じて提供される場合を除く)

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1
- sha224
- sha256
- sha384
- sha512

必須: はい

<KEY_FILTER>

キー参照 (例: `key-reference=0xabc`) または attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。プライベートキーのキー使用サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<DATA_TYPE>

データパラメータの値を、署名アルゴリズムの一部としてハッシュ化するかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

有効な値:

- raw
- ダイジェスト

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI で PureEdDSA メカニズムを使用して署名を生成する

> **Important**
>
> PureEdDSA 署名オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

CloudHSM CLI の crypto sign ed25519 コマンドを使用して、Ed25519 プライベートキーと PureEdDSA 署名メカニズムを使用して署名を生成します。PureEdDSA (Ed25519 とも呼ばれます) は、IETF ウェブサイトの RFC 8032、セクション 5.1.6 で定義されているように、プレハッシュなしで raw メッセージに直接署名します。

> **Note**
>
> PureEdDSA の最大メッセージサイズは 16,000 バイトです。大きなメッセージの場合は、を使用しますCloudHSM CLI で HashEdDSA メカニズムを使用して署名を生成する。

crypto sign ed25519 コマンドを使用するには、まず AWS CloudHSM クラスターに Ed25519 プライベートキーが必要です。`curve` パラメータを に設定`ed25519`し、`sign`属性を に設定して、CloudHSM CLI を使用して非対称 EC キーペアを生成する コマンドを使用して Ed25519 プライベートキーを生成できます`true`。

> **Note**
>
> 署名は、 CloudHSM CLI の暗号化検証カテゴリ サブコマンド AWS CloudHSM を使用してで検証できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

- PureEdDSA 署名オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。
- メッセージデータは 16,000 バイト以下にする必要があります。大きなメッセージの場合は、 を使用しますCloudHSM CLI で HashEdDSA メカニズムを使用して署名を生成する。

Syntax

```
aws-cloudhsm > help crypto sign ed25519
Sign with the Ed25519 mechanism

Usage: crypto sign ed25519 [OPTIONS] --key-filter [<KEY_FILTER>...] <--data-
path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --data-path <DATA_PATH>
          The path to the file containing the data to be signed
      --data <DATA>
          Base64 Encoded data to be signed
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
  -h, --help
          Print help
```

例

これらの例は、 crypto sign ed25519を使用して PureEdDSA 署名メカニズムを使用して署名を生成する方法を示しています。このコマンドは、事前ハッシュなしで raw メッセージに直接署名します。このコマンドは HSM で Ed25519 プライベートキーを使用します。

Example例: ベース 64 でエンコードされたデータの署名を生成する

```
aws-cloudhsm > crypto sign ed25519 \
    --key-filter attr.label=ed25519-private \
```

```
    --data YWJj
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000006c1507",
    "signature": "NjWz797ntYSLFwg7nKYYdn
+On3cCMj4zKz059wadVVlBHxyxe4JrSZxgekwb9AYR5xFxuVE9dTnDSo+gCaW/CQ=="
  }
}
```

Example例: データファイルの署名を生成する

```
aws-cloudhsm > crypto sign ed25519 \
    --key-filter attr.label=ed25519-private \
    --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000006c1507",
    "signature": "NjWz797ntYSLFwg7nKYYdn
+On3cCMj4zKz059wadVVlBHxyxe4JrSZxgekwb9AYR5xFxuVE9dTnDSo+gCaW/CQ=="
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。デコードされたデータは 16,000 バイト以下にする必要があります。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。ファイルの内容は 16,000 バイト以下にする必要があります。

必須: はい (データパラメータで指定されない限り)

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。プライベートキーのキー使用サービスのクォーラム値が 1 より大きい場合にのみ必要です。

必須: いいえ

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ
- CloudHSM CLI で PureEdDSA メカニズムで署名された署名を検証する
- CloudHSM CLI で HashEdDSA メカニズムを使用して署名を生成する

CloudHSM CLI で HashEdDSA メカニズムを使用して署名を生成する

> **Important**
>
> HashEdDSA 署名オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

CloudHSM CLI の crypto sign ed25519ph コマンドを使用して、Ed25519 プライベートキーと HashEdDSA 署名メカニズムを使用して署名を生成します。HashEdDSA の詳細については、NIST SP 186-5、セクション 7.8 を参照してください。

crypto sign ed25519ph コマンドを使用するには、まず AWS CloudHSM クラスターに Ed25519 プライベートキーが必要です。`curve` パラメータを に設定`ed25519`し、`sign`属性を に設定して、

CloudHSM CLI を使用して非対称 EC キーペアを生成する コマンドを使用して Ed25519 プライベートキーを生成できますtrue。

> **Note**
>
> 署名は、 CloudHSM CLI の暗号化検証カテゴリ サブコマンド AWS CloudHSM を使用して で検証できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。
- HashEdDSA 署名オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

Syntax

```
aws-cloudhsm > help crypto sign ed25519ph
Sign with the Ed25519ph mechanism

Usage: crypto sign ed25519ph [OPTIONS] --key-filter [<KEY_FILTER>...] --data-
type <DATA_TYPE> --hash-function <HASH_FUNCTION> <--data-path <DATA_PATH>|--
data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --data-path <DATA_PATH>
```

```
          The path to the file containing the data to be signed
      --data <DATA>
          Base64 Encoded data to be signed
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
      --data-type <DATA_TYPE>
          The type of data passed in, either raw or digest [possible values: raw,
 digest]
      --hash-function <HASH_FUNCTION>
          Hash function [possible values: sha512]
  -h, --help
          Print help
```

例

これらの例は、 crypto sign ed25519phを使用して Ed25519ph 署名メカニズムとsha512ハッシュ関数を使用して署名を生成する方法を示しています。このコマンドは HSM でプライベートキーを使用します。

Example例: ベース 64 でエンコードされたデータの署名を生成する

```
aws-cloudhsm > crypto sign ed25519ph \
    --key-filter attr.label=ed25519-private \
    --data-type raw \
    --hash-function sha512 \
    --data YWJj
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000401cdf",
    "signature": "mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h2OUmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg=="
  }
}
```

Example例: データファイルの署名を生成する

```
aws-cloudhsm > crypto sign ed25519ph \
    --key-filter attr.label=ed25519-private \
    --data-type raw \
    --hash-function sha512 \
    --data-path data.txt
```

```
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000401cdf",
    "signature": "mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h2OUmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg=="
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データパラメータで指定されない限り)

<HASH_FUNCTION>

ハッシュ関数を指定します。Ed25519ph は SHA512 のみをサポートします。

有効な値:

- sha512

必須: はい

<KEY_FILTER>

キー参照 (例: `key-reference=0xabc`) または attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。プライベートキーのキー使用サービスのクォーラム値が 1 より大きい場合にのみ必要です。

必須: いいえ

<DATA_TYPE>

データパラメータの値を、署名アルゴリズムの一部としてハッシュ化するかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

有効な値:

- raw
- ダイジェスト

必須: はい

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI で ML-DSA メカニズムを使用して署名を生成する

CloudHSM CLI の crypto sign ml-dsa コマンドを使用して、ML-DSA プライベートキーと ML-DSA 署名メカニズムを使用して署名を生成します。

crypto sign ml-dsa コマンドを使用するには、まず AWS CloudHSM クラスターに ML-DSA プライベートキーが必要です。ML-DSA プライベートキーは、 `sign` 属性を に設定して CloudHSM CLI を使用して非対称 ML-DSA キーペアを生成する コマンドを使用して生成できます`true`。

> **Note**
>
> 2026 年 9 月 1 日以降、ML-DSA は hsm2m.medium クラスターの FIPS モードで使用できます。

> **Note**
>
> CloudHSM CLI の暗号化検証カテゴリ サブコマンド AWS CloudHSM を使用して、 で署名を検証できます。

raw データまたは計算された外部 mu データの 2 つのデータ型を使用して署名を生成できます。

外部 mu を生成するステップ

`external-mu` データ型を使用するには、まず ML-DSA パブリックキーを使用してメッセージから 64 バイトの mu ダイジェストを生成する必要があります。

1. CloudHSM CLI を使用してパブリックキー PEM ファイルを生成します。

   ```
   aws-cloudhsm > key generate-file --encoding pem --path /tmp/public_key.pem --filter
    attr.label="mldsa-public-key-example"
   ```

   詳細については、「CloudHSM CLI で非対称キーをエクスポートする」を参照してください。
2. 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、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。
3. メッセージプレフィックスをメッセージに追加します。
4. SHAKE256 を使用してパブリックキーをハッシュし、ステップ 3 の結果に付加します。
5. 結果をハッシュして 64 バイトの外部 mu を生成します。

OpenSSL の例

次の例ではOpenSSL、3.5 を使用して外部 mu を作成します。

```
{
    openssl asn1parse -inform PEM -in public_key.pem -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 | base64
```

結果の base64 エンコードされた 64 バイトダイジェストを の--dataパラメータとして使用します--data-type external-mu。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help crypto sign ml-dsa
Signs data with a ML-DSA private key

Usage: crypto sign ml-dsa [OPTIONS] --key-filter <KEY_FILTER> --data-type <DATA_TYPE>
 --hash-function <HASH_FUNCTION>

Options:
      --key-filter <KEY_FILTER>
          The key reference or a list of key attributes to filter for a key to sign
 with

      --hash-function <HASH_FUNCTION>
          The hash function to use with ML-DSA. [possible values: shake256]

      --data <DATA>
          Data to be signed

      --data-path <DATA_PATH>
          Path to file containing data to be signed

      --data-type <DATA_TYPE>
          Specifies if the data is raw or has been processed using external mu.
 [possible values: raw, external-mu]

      --approval <APPROVAL>
```

```
          File path of signed quorum token file to approve operation. Necessary for
 keys with quorum enabled

  -h, --help
          Print help
```

例

例: ML-DSA プライベートキーを使用して raw データに署名する

このコマンドを使用して、ML-DSA プライベートキーで raw データに署名します。

```
aws-cloudhsm > crypto sign ml-dsa \
    --key-filter attr.label=ml-dsa-private-key-example \
    --data-type raw \
    --hash-function shake256 \
    --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000000f63",
    "signature": "+kO53l20YLjfUzDPbiJkFK/IWNUXobfqamJ7y2Tt7Nn3S6w3/h
+oPZm939ieNafVIwn1DsHyGm/
tLHFShZlFooi8alFyl76+VdXo5iJTmiBwYI5AK57ly8/2vj6zDRugnsLXbW40iMDlckWdZiwbYg/
yKfbHVFrYXywPkGbKk5d
+jZwvHlUAwnlc4MPAWKoauwkO86zlKjhuydihrMRBIPhUiXR21nkJ7dFMngzqHgBtwR08oPzMI5BLRpGL3euCFm816BWHzC
+Y5m0cPKbdjl/UTwFvh2UBytaqBu0wx6QcCsxsbZnBaEkspxYE62BuXZmM62B2VJsMG2j05MpaUULl6/
Yl2EeyqWEh3SqwAlN0QYRRqM4T3NkBwVfJ1JE5n4tgp91fL33iL4BoVeqStRJ0McA9+s
+EkYQ8UBM150CnHhqA47tMGmM0KfWPPx8H0xR7MqEaAAJup2D0SQfYfBbh4XKDVJpw1tpY/
hXlzEiWTTA6GVyYRVbsVGRDcxmyrzNf1DRSaHSgJksioSZ+xn0X8TKYnYp/
Fx80HnpO002RyQrzr+N8HvMemjVLhiY2eYYPQQsHSMZMZNbyG3SztiIYordUjc+6Ufwf
+fn9fbRzfBtkQJI7dJxgp518Z90FOzDigOq9p3S36W3K9RqUP9g3PeduucOm3YONdgl8TVpiuG0uC1JUUgUkh5yTNc49hPz
RKD3GVv6c2Fscj4KnL68j0N93aTCBpEWNv0hFWrIZJiRFinVx9hZPDeiFkomhMKR1+MjPeMY9hUE1lhOXUxbVSrtiMJBRjV
pi7pGKuNpm7VYbhRrbJ2hwPKiYgwkxhHf+SEoO5mkpzfQjADt5LOTUa+ZVlv5hWyp8v7YxDhV5Qv2jsII1h/
d1KX4A1BOrwS420/vT3haTKG1if9LphHziWdOFdRqJ40zt2MPaSZO9ARYN1Os1dqq
+Mf6dvzFHrcllcTrSlZvg2y31f0CP5IpjyV6mT3UjW0ccyIYTyaRTgcjJU7xj8qOHbEMbr8waGRSypNrBP4LJYCc6OJsljV
nGdGiPXCrQH6TH4ukV/dFaxueq9/WzFtHG6d6gQibytw1qrkYNSOv80cdyUNeZw6FJ5IlSvzM87N
+B18ghMo6J6yu/N6IJqNfyHtzRcgYtuo3O1bT5l+dvKoINw5ZXYW18ei/GXIVMn/z+TDZUl+TdUAUg6Q1C6zN
+TLX0Qdy+mirrjAvrrLT2aexOeVhqT8jgAcwQdZqnNg0nnp6IqNvepDtF1J5FnmRfkYCKxLXSXd/25q
+xeSMD3sVE9t8XwpjEyg9WGOLK7nBX8AgeeCFcYUyEIMq0sBZUNOeHWeY2R91Z6paj/W
+Is6FwHlP7dMe7zggiUxOk8oA8JWj9YlAI0KkijaBi047X78HJ/w5ywO5ybz663zBm7Cl7SOAARhGsPJwOSfbh/
U3Pq1ATNFv73O7dIoRY4E4czXIKHfKKkO1v4N5B6HneHlicCUAsz/
o78VuF36h5V29CsEL9qt0a01A1Q6jNigO4eTZ6BEqiRiM7LJoNjLrUn9ZK1czctr2hpjDXNi/
R+3zWwC1PqODNblqiQTV/2kxfjGIf+tIUsuPPPOzpRE3zgUrFcWBEpqBGIYMS0fBwN09Y
```

```
+RiE4jhtW+UM+rZB7lxSHUbAKme7Y5KRX9aIoRtmxJE5BnWnyeztZcQ02xBdTHSvXk
+fN2FkAU3rIPrsTaoTjmppmlwJoMaIO0DxSEsdGTpMgFwE
+B3Zc2StD/9zPCSHAC6lRHZL7BVqKOtm3Vc3CBzoH619HKu6jOteEDCRrNqfOzC8i7/
nv1GenadN6p7Ln+agE47/mOl2E/K254ULcl51eSbZpLpI+VDbN+yAFl40ErVxlWf
+YWg1RZziTnXCnHMWZOuJzqwsPuvaTCUTrh31MVSKvz0xGYZsepvsX8ho4xL003PaF9cOXnW9qgBryXb5JCQPbL8nFtxKcm
gPbDGOovkPwbAvRkYHjX6FbnysUqR7en0hAhiFsx0OZ90bRp1k5XYOW0YhC5degOoqpn9kTzbEnc58J9JnVq
+J5sHCaobp8yUXN6/cbM7l9+6G12oowTCbitkGOJ/
nZozeqnN8kdTw9HvVnOOMCD29ks38UEzFg1CaTgixLjjFMxgnN5twyK31HN0SCkcpmVkSnS/
oGZpMVqsfq5oIvN7+E3Ehi/KHHw52bhAgTiAq/ZTDpC8sVK0BwU2LgIXJUT/4dVTocOyRQb
+6fig4FaN4f9MGzB0cYiGKCACCZ+X7dtFgHzxqx9/
Qo4AQkmRODUiYs8STXKUF7igDBQsBEfPxsU6W24IgXrTz5JnIL9x/r2l2zz82ei2DWCxS4qlymz0Zs5H/
gjFsazYFbM8fYOEpCaB9vtuMSzVDILFFkSRVJrSAloG2vFu8fcjK5BA7a2h3uhBfaftmJFKiaEsCal
+mw0PzhDaMsuLgXXEm+RXuF1t8Bpiv+d47VjMV6sC6FaNi1gY+eH7/
oj1D3KC2xG2aSZjgdyhEm0uXLhCpRz1X7bpfELcxs1FqAGMLJFYGSOmUlrqrL4WROQT4FJ
+iZ0UFbqcujetJuWv9z+kBf9s/5r3ZoGd5VXNnNKQz7BV/
+wxjPk9Ui184ZHOOL2xM7BStdEDgTuMAbUsI5qDMCIE77g1JqZGWpeRuFf/
voJ2TNM7yUEikvf52UOihQbdiT6+feaK9urR4wcPDIwDB0/p8tObmVOCmKbhzVtXsnAjylpSk/
YHQ4fj6cxsMKSVz9IZihpga/DX12ki7Oacj/AJeGFIZYjBo/Kr2iEtoq5b0YQfC8Lp
+CItAelNzFzJ5IXbhmC8WtuhIk6ylQP5xyPofdlOJ1KBw46QckdxnMl3asO7rI4lHj2oXB4QiKVwgCbmvTw5U5O7NFB3rbp
UGTy7MHhf2hT9CbdJfxUUMXuv30nqc9xxkNxDozRw1KEAaG
+Ng/4p3WGmLRcSFj1JUHODkJyrubrD0djb5f4gNklKe4aNvsDF0tff4AUUQ0d/gYaLpdLr8/
sREhsfIzSgqKmyv9bi9QAAAAAAAAAAAAAAAAAAAAAAAAAABIgLTs="
  }
}
```

例: ML-DSA プライベートキーを使用して外部 mu データに署名する

このコマンドを使用して、ML-DSA プライベートキーで外部 mu データに署名します。

```
aws-cloudhsm > crypto sign ml-dsa \
    --key-filter attr.label="MLDSA_private_key" \
    --data-type external-mu \
    --hash-function shake256 \
    --data DgFbkWgb9fbejQUSbJ09/
Nmp7phb714RTkWdNAwQqd9u7M7XCWbI8EoaIeGcZCUwIzaG6kDavZDYxCcaEicbTw==
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000000be2",
    "signature": "jxSKx9W/5Tu0lFySyrJPpNo4v/
tcNGfeC4Zt/0ubAAt7IQxH8MEjwcA7OO9AyaQntxDpzua8m+m5fEwiVrwmH
+zZhgw2EvXrFDU7zz6n6RRUI2RyXG6siistsBPwDZUU1G8oMp33k3E4LEL1al1AaYDncO9fQx1nS5wjg33OxzQbZi7GqRKB
+1MCPlSC3TshojRXlg2ELx32rRdzGhWAGZ/
A8qa9FIcS6LZlF35NZiJbK3W2mIFl8U5S2ZMXgIdfqm0wSNDibB4p23AR4SVz1Wghbu1To0pbeeaz74LxbZ3CIhpW
```

```
+UmtF8VZfGw/AbOQK2K1Z27VQuIc2Ru3L6PpSuDl4LM96ZdGPW6n/xjBMax83pAz9qwm
+cy/yjD28GCUKqwzNrjxB49ak6RGCYiUjCFG9UfEQ46xmJjOhvGaepf9XqjqmcO66rk
+mGThL96CGOpz1+QI91OQRSu3bDV9g2+Zl
+vEArvTUPLV5nbqZ598Uppswfg94n248S1P8M/5vkjiYBUD8asNcuQp2lQBXL4iDpuazewUyTLy2GADbwlfEYwxQ
+xELWxE4Gv8HMiQX2x97pC2BNjxlN1z4wMoPqsVlw
+G5xS2juTepUW3G9O0jgHvIVLuq2qC0XhSPRAvxWBE3iFVTDXwlpfaqjdRuhp5UwUfrL6m
+jKuLDRFxtzGFUmUfHtImSoZ0oNiQl+45Tlf+AZsYJno4chx5lQkVca7KI8jkvDLRqoxZFPb3T/
GlHoQ1R2y9EzB6LI0dCSAGWiU1kRDXcoXYls4Q6JfDZW6TRj0IAo0StVT7Au1S
+N+jPPKfpU7lJfROJglbt6gQM7rGy8naKsfEfBij+ufd89iKC+qwDP2WSx
+XsLMzfBBWHA8hJMGQHwEHeMvrjTJaSgKSP0M4pdcAV8g5A5TIaLgic4pszSVmc2PDmGw6ssAvS2M3SbnGCxUIcC7ziI74X
dy/uiZ8CirYhcvCH7jUsWR4J2JO+vAjnQJZZRXmUmmh3HHQoihV/P2kpmM8YxIVein1+
+gizJC9eijEuDh4rTr3SinO8vMoiclUG0+ah3yEKF6LjqtbWnNHY8Hnvm2w22yX1jjK4fsspsc9XbS3Og9un8qkaSpLphtz
f3CYiGLLFKupYjAKckU5J6t6LEdP8xoQF3alGVo/2lGQGIJrlg/qS2/7XpK5feeERtDxTCVZZhFj2w27OMsLTX/
Al0Koz8KCQFk5OWjQEfY3yicwP/+JB/ftQ
+5Uhp3CmOqSqNU7hRgL5e6QYi1E3D1KB8Je35Lz2PuzuClTttgWaExXe747fPvAbu3OJEjApdCYkaj0VokYhk2LH
+icpsCQfnMQjddei8WRyPMehp64PJRAXcCuX9qPiwfeCNxjplG+0PnY1SYKVRCQdbuuWORrAbLv8q7wSpLVB2A/
R0MISoZT6IGLiLxKbh7KZ7cSCV0rypD1+2Z7lYbvaUztP2CniJzqOuCbMDhlFzelbQaxY9UBlCiWjpxj7WShBnJuDD23Lhj
++W3mpaYBhUDMGMX0EWluSF7q2hAH8HcfMOUYDjaoV
+MoJupVb1yA6+NM5b6PmZR7RPiGPVBpoRR11GUAI9icFfYOr6gsc5yJZDsGtURz1PzpnSb21ZPlib9tHRW88kl
+2GijZes7DvyHhyavvfeO5M83uTCaYuohrVxKMJVewpW+O6nnPDxi
+dlWZ0NRQoU1CibRQiQi3ExDpmdQWMHHjkqVxxx6cnvn7wH60ztq5lsu3FoZ5gJSV8qabs3yjeh/
Tv5KlA7CaSVqcsYkqMEU2aT40Nysk2PO3v5b8ZmR9uU6YbQx7une+RZcmDhYXKtTG6NFaSNJjU3RwL805vaX/
hTVBc5o4jNK40JNIo1efrLzKp0V/cYk6/uBAoDAS4dbVhrf1GAyqlAbTc52dJJdJhzYaBJxddcdT
+CJAS8J1omVcPeMKn+Ny+Fy2sR53fHbP2E7OExbfYxToktPC4BRucIeFKWoYHYTSsYnennIbZBBfT4rPc7cA/
lWP94oxgjpCwbwQykLXeHh+MOgEDkk/Z+Gw/
xweTOWp6reiGUhERkXtniK2f3qjcRT03b9K6wNA7M08AKDSEtm3jT9KHhlJH9Tb1wOpPhUmj2MbIvbyG2yZknmiDIIwQmhM
PuG1yZMn9z+4tKFk6PYwbZON1GPPsVmhgKp+8AVIyLKVOoDNHX4io0u+Xk3+hAu5qU3kEJV29crhOCCJ8G2tBF
+y0Ho7S3ECAhGqFDxEKEQUFjXvU0gr8yGrqvgP1yC9ZxCu6TR5EESKtJc/
X2Acqyjg2Hx1IABe8ztmPT79FJyiAKDz5XsIfgBp0/
b1gBc3QV0VjYx4SvuYlVB0bFHqQmfs1bD2Ah9iCqPCe3qQli8q1T677shw07ijbyAYD6rfZqUDTwZr6EAQMC90v9sSaQY/5
FCquY2jZb5fTlDBk6TLYB8mMN2DJP+cDJ0FfhHOrt9/QIvrud2ZoldhAFPAFA841AYWb+BDVcZmLsIh
+RrFS0+vqyNyGN3TvOdxXxobFYuXA3QfcjcM6zMG+zcntK4b552QwKhVRzD9cvGB4AUM0ab0ivF
+8sfZVKtTYHepP/
FUiXk0QnMBpwydVrw0drz3dYRmvVHLSpJJemSVrT4hYmBrN0nMPWiiz9fANBHHC6F0TOwzeqmlrpXPXIiWOAp8ox/
fF9HcrULwkqJ5Bu6GVwXSCe19nFWVjt+8FPaa2B3mRaBgz71OLcpG4/LUD0rF0lUBvebl/
r8sEE5K2lhuak6Y/rg4/brTwVrFZdaYcmBUOSmboUpvox08HLY57XhJe8YoCQGQUezkL
+ZhXGMaYmphxmEp/4aWs6PvOfOWh849FCfQp9aEqfiX3d1rqNFK6NkLBSNEYf2D
+ikY/dLW98XUqDPHBipSlLY926ThO1Fc+BUFeeR9g6czyiB3K2Dcgi70cb61kAQ/
hHWNaaCdWfOdC7swjOlSlTMBo7CF6K
+FMA2uM9+8ZGOdlQOQUEgSutETmG9vNraGhC3qSZyhJBRtiDLSCxEQYu9BpgDDiAkK3ySpqu1urvD0uXs8fL5CiErOkBcZm
  }
}
```

例: ファイルパスからデータに署名する

このコマンドを使用して、ML-DSA プライベートキーを使用してファイルからデータに署名します。

```
aws-cloudhsm > crypto sign ml-dsa \
    --key-filter attr.label=ml-dsa-private-key-example \
    --data-type raw \
    --hash-function shake256 \
    --data-path /tmp/data.bin
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000000f63",
    "signature": "..."
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データを通じて提供される場合を除く)

<HASH_FUNCTION>

ML-DSA で使用するハッシュ関数を指定します。

有効な値:

- シェイク256

必須: はい

<KEY_FILTER>

キーリファレンス (などkey-reference=0xabc) または attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のスペース区切りリスト。一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。プライベートキーのキー使用サービスのクォーラム値が 1 より大きい場合にのみ必要です。

必須: いいえ

<DATA_TYPE>

提供されたデータのデータ型を指定します。ハッシュされていないデータrawには を使用し、計算された外部 mu データexternal-muには を使用します。

有効な値:

- raw
- external-mu

必須: はい

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI で ML-DSA メカニズムで署名された署名を検証する
- CloudHSM CLI を使用して非対称 ML-DSA キーペアを生成する

CloudHSM CLI で RSA-PKCS メカニズムを使用して署名を生成する

CloudHSM CLI の crypto sign rsa-pkcs コマンドを使用して、RSA プライベートキーと RSA-PKCS 署名メカニズムを使用して署名を生成します。

crypto sign rsa-pkcs コマンドを使用するには、まず AWS CloudHSM クラスターに RSA プライベートキーが必要です。sign 属性を true に設定して、CloudHSM CLI で非対称 RSA キーペアを生成する コマンドを使用して RSA プライベートキーを生成できます。

> **Note**
>
> 署名は、 CloudHSM CLI の暗号化検証カテゴリ サブコマンド AWS CloudHSM を使用して で検証できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help crypto sign rsa-pkcs
Sign with the RSA-PKCS mechanism

Usage: crypto sign rsa-pkcs --key-filter [<KEY_FILTER>>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be signed
```

```
      --data <DATA>
          Base64 Encoded data to be signed
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
      --data-type <DATA_TYPE>
          The type of data passed in, either raw or digest [possible values: raw,
 digest]
  -h, --help
          Print help
```

例

これらの例は、crypto sign rsa-pkcs を使用して、RSA-PKCS 署名メカニズムと SHA256 ハッシュ関数を使用して署名を生成する方法を示しています。このコマンドは HSM でプライベートキーを使用します。

Example例: ベース 64 でエンコードされたデータの署名を生成する

```
aws-cloudhsm > crypto sign rsa-pkcs \
    --key-filter attr.label=rsa-private \
    --hash-function sha256 \
    --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==="
  }
}
```

Example例: データファイルの署名を生成する

```
aws-cloudhsm > crypto sign rsa-pkcs \
    --key-filter attr.label=rsa-private \
    --hash-function sha256 \
    --data-path data.txt
{
```

```
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==""
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データを通じて提供される場合を除く)

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1
- sha224
- sha256
- sha384
- sha512

必須: はい

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。プライベートキーのキー使用サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<DATA_TYPE>

データパラメータの値を、署名アルゴリズムの一部としてハッシュ化するかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

RSA-PKCS の場合、データは RFC 8017、セクション 9.2 で規定されている DER エンコード形式で渡す必要があります。

有効な値:

- raw
- ダイジェスト

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI で RSA-PKCS-PSS メカニズムを使用して署名を生成する

CloudHSM CLI の crypto sign rsa-pkcs-pss コマンドを使用して、RSA プライベートキーと `RSA-PKCS-PSS` 署名メカニズムを使用して署名を生成します。

crypto sign rsa-pkcs-pss コマンドを使用するには、まず AWS CloudHSM クラスターに RSA プライベートキーが必要です。sign 属性を true に設定して、CloudHSM CLI で非対称 RSA キーペアを生成する コマンドを使用して RSA プライベートキーを生成できます。

> **Note**
>
> 署名は、 CloudHSM CLI の暗号化検証カテゴリ サブコマンド AWS CloudHSM を使用して で検証できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help crypto sign rsa-pkcs-pss
Sign with the RSA-PKCS-PSS mechanism

Usage: crypto sign rsa-pkcs-pss [OPTIONS] --key-filter [<KEY_FILTER>...] --
hash-function <HASH_FUNCTION> --mgf <MGF> --salt-length <SALT_LENGTH> <--data-
path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>        Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]   Key reference (e.g. key-
reference=0xabc) or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key
      --hash-function <HASH_FUNCTION>  [possible values: sha1, sha224, sha256, sha384,
 sha512]
      --data-path <DATA_PATH>          The path to the file containing the data to be
 signed
      --data <DATA>                    Base64 Encoded data to be signed
```

```
      --mgf <MGF>                          The mask generation function [possible values:
 mgf1-sha1, mgf1-sha224, mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --salt-length <SALT_LENGTH>      The salt length
      --approval <APPROVAL>            Filepath of signed quorum token file to approve
 operation
      --data-type <DATA_TYPE>          The type of data passed in, either raw or digest
 [possible values: raw, digest]
  -h, --help                           Print help
```

例

これらの例は、crypto sign rsa-pkcs-pss を使用して、RSA-PKCS-PSS 署名メカニズムと SHA256 ハッシュ関数を使用して署名を生成する方法を示しています。このコマンドは HSM でプライベートキーを使用します。

Example例: ベース 64 でエンコードされたデータの署名を生成する

```
aws-cloudhsm > crypto sign rsa-pkcs-pss \
    --key-filter attr.label=rsa-private \
    --hash-function sha256 \
    --data YWJjMTIz \
    --salt-length 10 \
    --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

Example例: データファイルの署名を生成する

```
aws-cloudhsm > crypto sign rsa-pkcs-pss \
    --key-filter attr.label=rsa-private \
    --hash-function sha256 \
    --data-path data.txt \
    --salt-length 10 \
```

```
    --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データを通じて提供される場合を除く)

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1
- sha224
- sha256
- sha384

- sha512

必須: はい

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<MGF>

マスク生成関数を指定します。

> **Note**
>
> マスク生成関数のハッシュ関数は、署名メカニズムのハッシュ関数と一致する必要があります。

有効な値:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必須: はい

<SALT_LENGTH>

ソルトの長さを指定します。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。プライベートキーのキー使用サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<DATA_TYPE>

データパラメータの値を、署名アルゴリズムの一部としてハッシュ化するかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

有効な値:

- raw
- ダイジェスト

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

関連トピック

- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI では、crypto verify はコマンドグループの親カテゴリであり、親カテゴリと組み合わせるとファイルが指定されたキーによって署名されているかどうかを確認します。crypto verify には次のサブコマンドがあります。

- crypto verify ecdsa
- 暗号化検証 ed25519
- 暗号化検証 ed25519ph
- 暗号化検証 ml-dsa
- crypto verify rsa-pkcs
- crypto verify rsa-pkcs-pss

crypto verify コマンドは、署名されたファイルをソースファイルと比較し、両者が指定されたパブリックキーと署名メカニズムに基づいて暗号的に関連するかどうかを分析します。

> **Note**
>
> ファイルは CloudHSM CLI の暗号化署名カテゴリオペレーション AWS CloudHSM でサインインできます。

CloudHSM CLI で ECDSA メカニズムを使用して署名された署名を検証する

CloudHSM CLI の crypto verify ecdsa コマンドを使用して、次のオペレーションを完了します。

- 指定されたパブリックキーによって HSM でファイルが署名されていることを確認します。
- ECDSA 署名メカニズムを使用して署名が生成されたことを検証します。
- 署名されたファイルをソースファイルと比較し、両者が指定された ecdsa パブリックキーと署名メカニズムに基づいて暗号的に関連するかどうかを判断します。
- ECDSA 検証関数は、r と のコンポーネントが raw バイナリデータとして連結される `r||s`形式の署名を想定しています。

crypto verify ecdsa コマンドを使用するには、まず AWS CloudHSM クラスターに EC パブリックキーが必要です。`verify` 属性を `true` に設定して CloudHSM CLI で PEM 形式キーをインポートする コマンドを使用して、EC パブリックキーをインポートできます。

> **Note**
>
> CloudHSM CLI の CloudHSM CLI の暗号化署名カテゴリ サブコマンドを使用して署名を生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help crypto verify ecdsa
Verify with the ECDSA mechanism

Usage: crypto verify ecdsa --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-
path <SIGNATURE_PATH>|--signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
      --data-type <DATA_TYPE>
          The type of data passed in, either raw or digest [possible values: raw,
 digest]
  -h, --help
          Print help
```

例

これらの例は、crypto verify ecdsa を使用して、ECDSA 署名メカニズムと SHA256 ハッシュ関数を使用して生成された署名を検証する方法を示しています。このコマンドは HSM でパブリックキーを使用します。

Example例: Base64 でエンコードされた署名を Base64 でエンコードされたデータで検証する

```
aws-cloudhsm > crypto verify ecdsa \
    --hash-function sha256 \
    --key-filter attr.label=ec-public \
    --data YWJjMTIz \
    --signature 4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: データファイルを使用して署名ファイルを検証する

```
aws-cloudhsm > crypto verify ecdsa \
    --hash-function sha256 \
    --key-filter attr.label=ec-public \
    --data-path data.txt \
    --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: 偽の署名関係を証明する

このコマンドは、/home/data にあるデータが、ラベル ecdsa-public 付きのパブリックキーによって、/home/signature にある署名を生成する ECDSA 署名メカニズムを使用して署名されたかどうかを検証します。指定の引数が真の署名関係を構成していないため、コマンドは、エラーメッセージを返します。

```
aws-cloudhsm > crypto verify ecdsa \
    --hash-function sha256 \
    --key-filter attr.label=ec-public \
    --data aW52YWxpZA== \
    --signature +ogk7M7S3iTqFg3SndJfd91dZFr5Qo6YixJl8JwcvqqVgsVuO6o+VKvTRjz0/
V05kf3JJbBLr87Q+wLWcMAJfA==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データパスを通じて提供される場合を除く)

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1
- sha224
- sha256
- sha384
- sha512

必須: はい

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<SIGNATURE>

Base64 でエンコードされた署名。

必須: はい (署名パスを通じて提供される場合を除く)

<SIGNATURE_PATH>

署名の場所を指定します。

必須: はい (署名パスを通じて提供される場合を除く)

<DATA_TYPE>

データパラメータの値を、署名アルゴリズムの一部としてハッシュ化するかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

有効な値:

- raw
- ダイジェスト

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI で PureEdDSA メカニズムで署名された署名を検証する

> **Important**
>
> PureEdDSA 署名検証オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

CloudHSM CLI の crypto verify ed25519 コマンドを使用して、次のオペレーションを完了します。

- 特定の Ed25519 パブリックキーを使用して、データまたはファイルの署名を検証します。
- 署名が PureEdDSA 署名メカニズム (Ed25519) を使用して生成されたことを確認します。

> **Note**
>
> PureEdDSA の最大メッセージサイズは 16,000 バイトです。大きなメッセージの場合は、を使用しますCloudHSM CLI で HashEdDSA メカニズムで署名された署名を検証する。

crypto verify ed25519 コマンドを使用するには、まず AWS CloudHSM クラスターに Ed25519 パブリックキーが必要です。`curve` パラメータを に設定`ed25519`し、 `verify` 属性を に設定して CloudHSM CLI を使用して非対称 EC キーペアを生成する コマンドを使用して Ed25519 キーペアを生成するか`true`、 `verify` 属性を に設定して CloudHSM CLI で PEM 形式キーをインポートする コマンドを使用して Ed25519 パブリックキーをインポートできます`true`。

> **Note**
>
> CloudHSM CLI の CloudHSM CLI の暗号化署名カテゴリ サブコマンドを使用して署名を生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。
- PureEdDSA 署名検証オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。
- メッセージデータは 16,000 バイト以下にする必要があります。

Syntax

```
aws-cloudhsm > help crypto verify ed25519
Verify with the Ed25519 mechanism

Usage: crypto verify ed25519 [OPTIONS] --key-filter [<KEY_FILTER>...] <--
data-path <DATA_PATH>|--data <DATA>> <--signature-path <SIGNATURE_PATH>|--
signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
  -h, --help
          Print help
```

例

これらの例は、 crypto verify ed25519を使用して、PureEdDSA 署名メカニズムを使用して生成された署名を検証する方法を示しています。このコマンドは HSM で Ed25519 パブリックキーを使用します。

Example例: Base64 でエンコードされた署名を Base64 でエンコードされたデータで検証する

```
aws-cloudhsm > crypto verify ed25519 \
    --key-filter attr.label=ed25519-public \
    --data YWJj \
    --signature NjWz797ntYSLFwg7nKYYdn
+On3cCMj4zKz059wadVVlBHxyxe4JrSZxgekwb9AYR5xFxuVE9dTnDSo+gCaW/CQ==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: データファイルを使用して署名ファイルを検証する

```
aws-cloudhsm > crypto verify ed25519 \
    --key-filter attr.label=ed25519-public \
    --data-path data.txt \
    --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: 誤った署名関係を証明する

この例では、データ、署名、またはキーが一致しない場合、検証が失敗することを示しています。

```
aws-cloudhsm > crypto verify ed25519 \
    --key-filter attr.label=ed25519-public \
    --data bWVzc2FnZQ== \
    --signature NjWz797ntYSLFwg7nKYYdn
+On3cCMj4zKz059wadVVlBHxyxe4JrSZxgekwb9AYR5xFxuVE9dTnDSo+gCaW/CQ==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

検証する Base64 エンコードされたデータ。デコードされたデータは 16,000 バイト以下にする必要があります。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

検証するデータの場所を指定します。ファイルの内容は 16,000 バイト以下にする必要があります。

必須: はい (データパラメータで指定されない限り)

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: はい

<SIGNATURE>

Base64 でエンコードされた署名。

必須: はい (署名パスを通じて提供される場合を除く)

<SIGNATURE_PATH>

署名の場所を指定します。

必須: はい (署名パラメータで指定されない限り)

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ
- CloudHSM CLI で PureEdDSA メカニズムを使用して署名を生成する
- CloudHSM CLI で HashEdDSA メカニズムで署名された署名を検証する

CloudHSM CLI で HashEdDSA メカニズムで署名された署名を検証する

> **Important**
>
> HashEdDSA 署名検証オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされます。

CloudHSM CLI の crypto verify ed25519ph コマンドを使用して、次のオペレーションを完了します。

- 特定の Ed25519 パブリックキーを使用して、データまたはファイルの署名を検証します。
- HashEdDSA 署名メカニズムを使用して署名が生成されたことを確認します。HashEdDSA の詳細については、NIST SP 186-5、セクション 7.8 を参照してください。

crypto verify ed25519ph コマンドを使用するには、まず AWS CloudHSM クラスターに Ed25519 パブリックキーが必要です。`curve` パラメータを に設定`ed25519`し、 `verify` 属性を に設定して CloudHSM CLI を使用して非対称 EC キーペアを生成する コマンドを使用して Ed25519 キーペアを生成するか`true`、 `verify` 属性を に設定して CloudHSM CLI で PEM 形式キーをインポートする コマンドを使用して Ed25519 パブリックキーをインポートできます`true`。

> **Note**
>
> CloudHSM CLI の CloudHSM CLI の暗号化署名カテゴリ サブコマンドを使用して署名を生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。
- HashEdDSA 署名検証オペレーションは、非 FIPS モードの hsm2m.medium インスタンスでのみサポートされます。

Syntax

```
aws-cloudhsm > help crypto verify ed25519ph
Verify with the Ed25519ph mechanism

Usage: crypto verify ed25519ph [OPTIONS] --key-filter [<KEY_FILTER>...] --data-
type <DATA_TYPE> --hash-function <HASH_FUNCTION> <--data-path <DATA_PATH>|--
data <DATA>> <--signature-path <SIGNATURE_PATH>|--signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
      --data-type <DATA_TYPE>
          The type of data passed in, either raw or digest [possible values: raw,
 digest]
      --hash-function <HASH_FUNCTION>
          Hash function [possible values: sha512]
  -h, --help
          Print help
```

例

これらの例は、 crypto verify ed25519phを使用して Ed25519ph 署名メカニズムとsha512ハッシュ関数を使用して生成された署名を検証する方法を示しています。このコマンドは HSM で Ed25519 パブリックキーを使用します。

Example例: Base64 でエンコードされた署名を Base64 でエンコードされたデータで検証する

```
aws-cloudhsm > crypto verify ed25519ph \
    --hash-function sha512 \
    --key-filter attr.label=ed25519-public \
    --data-type raw \
    --data YWJj \
    --signature mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h2OUmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: データファイルを使用して署名ファイルを検証する

```
aws-cloudhsm > crypto verify ed25519ph \
    --hash-function sha512 \
    --key-filter attr.label=ed25519-public \
    --data-type raw \
    --data-path data.txt \
    --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

検証する Base64 エンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

検証するデータの場所を指定します。

必須: はい (データパラメータで指定されない限り)

<HASH_FUNCTION>

ハッシュ関数を指定します。Ed25519ph は SHA512 のみをサポートします。

有効な値:

- sha512

必須: はい

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「」を参照してくださいCloudHSM CLI のキー属性。

必須: はい

<SIGNATURE>

Base64 でエンコードされた署名。

必須: はい (署名パスを通じて提供される場合を除く)

<SIGNATURE_PATH>

署名の場所を指定します。

必須: はい (署名パラメータで指定されない限り)

<DATA_TYPE>

データパラメータの値を検証アルゴリズムの一部としてハッシュするかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

有効な値:

- raw
- ダイジェスト

必須: はい

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ
- CloudHSM CLI で HashEdDSA メカニズムを使用して署名を生成する

CloudHSM CLI で ML-DSA メカニズムで署名された署名を検証する

CloudHSM CLI の crypto verify ml-dsa コマンドを使用して、ML-DSA-based署名後にデータの整合性と信頼性を確認します。このコマンドを使用すると、次のことができます。

- 特定のパブリックキーが HSM 内のファイルに署名したことを確認します。
- ML-DSA 署名メカニズムが署名を生成したことを確認します。
- 署名付きファイルをソースファイルと比較して、特定の ML-DSA パブリックキーに基づいて暗号的に関連しているかどうかを判断します。

crypto verify ml-dsa コマンドを使用するには、まず `verify` 属性を に設定して AWS CloudHSM 、クラスターに ML-DSA パブリックキーが必要です`true`。

> **Note**
>
> 2026 年 9 月 1 日以降、ML-DSA は hsm2m.medium クラスターの FIPS モードで使用できます。

> **Note**
>
> CloudHSM CLI の CloudHSM CLI の暗号化署名カテゴリ サブコマンドを使用して署名を生成できます。

> **Note**
>
> `external-mu` データ型を使用するには、まず 64 バイトの mu ダイジェストを生成する必要があります。ステップについては、ml-dsa ドキュメントの「外部 mu を生成するステップ」を参照してください。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help crypto verify ml-dsa
Verifies a signature with a ML-DSA public key

Usage: crypto verify ml-dsa [OPTIONS] --key-filter <KEY_FILTER> --data-type <DATA_TYPE>
 --signature <SIGNATURE> --hash-function <HASH_FUNCTION>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --key-filter <KEY_FILTER>
          The key reference or a list of key attributes to filter for a key to verify
 with
```

```
      --hash-function <HASH_FUNCTION>
          The hash function to use with ML-DSA. [possible values: shake256]

      --data <DATA>
          Data that was signed

      --data-path <DATA_PATH>
          Path to file containing data that was signed

      --data-type <DATA_TYPE>
          Specifies if the data is raw or processed using external mu [possible values:
 raw, external-mu]

      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located

      --signature <SIGNATURE>
          Base64 encoded signature to be verified

  -h, --help
          Print help
```

例

これらの例は、 crypto verify ml-dsa を使用して ML-DSA 署名メカニズムが生成した署名を検証する方法を示しています。このコマンドは HSM でパブリックキーを使用します。

Example例: raw データを使用して署名を検証する

このコマンドを使用して、ML-DSA パブリックキーを使用して raw データの署名を検証します。

```
aws-cloudhsm > crypto verify ml-dsa \
    --key-filter attr.label=ml-dsa-public-key-example \
    --data-type raw \
    --hash-function shake256 \
    --data YWJjMTIz \
    --signature "+kO53l20YLjfUzDPbiJkFK/IWNUXobfqamJ7y2Tt7Nn3S6w3/h
+oPZm939ieNafVIwn1DsHyGm/
tLHFShZlFooi8alFyl76+VdXo5iJTmiBwYI5AK57ly8/2vj6zDRugnsLXbW40iMDlckWdZiwbYg/
yKfbHVFrYXywPkGbKk5d
+jZwvHlUAwnlc4MPAWKoauwkO86zlKjhuydihrMRBIPhUiXR21nkJ7dFMngzqHgBtwR08oPzMI5BLRpGL3euCFm816BWHzC
+Y5m0cPKbdjl/UTwFvh2UBytaqBu0wx6QcCsxsbZnBaEkspxYE62BuXZmM62B2VJsMG2j05MpaUUL16/
Yl2EeyqWEh3SqwAlN0QYRRqM4T3NkBwVfJ1JE5n4tgp91fL33iL4BoVeqStRJ0McA9+s
```

```
+EkYQ8UBM150CnHhqA47tMGmM0KfWPPx8H0xR7MqEaAAJup2D0SQfYfBbh4XKDVJpw1tpY/
hXlzEiWTTA6GVyYRVbsVGRDcxmyrzNf1DRSaHSgJksioSZ+xn0X8TKYnYp/
Fx80HnpO002RyQrzr+N8HvMemjVLhiY2eYYPQQsHSMZMZNbyG3SztiIYordUjc+6Ufwf
+fn9fbRzfBtkQJI7dJxgp518Z90FOzDigOq9p3S36W3K9RqUP9g3PeduucOm3YONdgl8TVpiuG0uC1JUUgUkh5yTNc49hPz
RKD3GVv6c2Fscj4KnL68j0N93aTCBpEWNv0hFWrIZJiRFinVx9hZPDeiFkomhMKR1+MjPeMY9hUE1lhOXUxbVSrtiMJBRjV
pi7pGKuNpm7VYbhRrbJ2hwPKiYgwkxhHf+SEoO5mkpzfQjADt5LOTUa+ZVlv5hWyp8v7YxDhV5Qv2jsII1h/
d1KX4A1BOrwS420/vT3haTKG1if9LphHziWdOFdRqJ40zt2MPaSZO9ARYN1Os1dqq
+Mf6dvzFHrcllcTrSlZvg2y31f0CP5IpjyV6mT3UjW0ccyIYTyaRTgcjJU7xj8qOHbEMbr8waGRSypNrBP4LJYCc6OJsljV
nGdGiPXCrQH6TH4ukV/dFaxueq9/WzFtHG6d6gQibytw1qrkYNSOv80cdyUNeZw6FJ5IlSvzM87N
+B18ghMo6J6yu/N6IJqNfyHtzRcgYtuo3O1bT5l+dvKoINw5ZXYW18ei/GXIVMn/z+TDZUl+TdUAUg6Q1C6zN
+TLX0Qdy+mirrjAvrrLT2aexOeVhqT8jgAcwQdZqnNg0nnp6IqNvepDtF1J5FnmRfkYCKxLXSXd/25q
+xeSMD3sVE9t8XwpjEyg9WGOLK7nBX8AgeeCFcYUyEIMq0sBZUNOeHWeY2R91Z6paj/W
+Is6FwHlP7dMe7zggiUxOk8oA8JWj9YlAI0KkijaBi047X78HJ/w5ywO5ybz663zBm7Cl7SOAARhGsPJwOSfbh/
U3Pq1ATNFv73O7dIoRY4E4czXIKHfKKkO1v4N5B6HneHlicCUAsz/
o78VuF36h5V29CsEL9qt0a01A1Q6jNigO4eTZ6BEqiRiM7LJoNjLrUn9ZK1czctr2hpjDXNi/
R+3zWwC1PqODNblqiQTV/2kxfjGIf+tIUsuPPPOzpRE3zgUrFcWBEpqBGIYMS0fBwN09Y
+RiE4jhtW+UM+rZB7lxSHUbAKme7Y5KRX9aIoRtmxJE5BnWnyeztZcQ02xBdTHSvXk
+fN2FkAU3rIPrsTaoTjmppmlwJoMaIO0DxSEsdGTpMgFwE
+B3Zc2StD/9zPCSHAC6lRHZL7BVqKOtm3Vc3CBzoH619HKu6jOteEDCRrNqfOzC8i7/
nv1GenadN6p7Ln+agE47/mOl2E/K254ULcl51eSbZpLpI+VDbN+yAFl40ErVxlWf
+YWg1RZziTnXCnHMWZOuJzqwsPuvaTCUTrh31MVSKvz0xGYZsepvsX8ho4xL003PaF9cOXnW9qgBryXb5JCQPbL8nFtxKcm
gPbDGOovkPwbAvRkYHjX6FbnysUqR7en0hAhiFsx0OZ90bRp1k5XYOW0YhC5degOoqpn9kTzbEnc58J9JnVq
+J5sHCaobp8yUXN6/cbM7l9+6G12oowTCbitkGOJ/
nZozeqnN8kdTw9HvVnOOMCD29ks38UEzFg1CaTgixLjjFMxgnN5twyK31HN0SCkcpmVkSnS/
oGZpMVqsfq5oIvN7+E3Ehi/KHHw52bhAgTiAq/ZTDpC8sVK0BwU2LgIXJUT/4dVTocOyRQb
+6fig4FaN4f9MGzB0cYiGKCACCZ+X7dtFgHzxqx9/
Qo4AQkmRODUiYs8STXKUF7igDBQsBEfPxsU6W24IgXrTz5JnIL9x/r2l2zz82ei2DWCxS4qlymz0Zs5H/
gjFsazYFbM8fYOEpCaB9vtuMSzVDILFFkSRVJrSAloG2vFu8fcjK5BA7a2h3uhBfaftmJFKiaEsCal
+mw0PzhDaMsuLgXXEm+RXuF1t8Bpiv+d47VjMV6sC6FaNi1gY+eH7/
oj1D3KC2xG2aSZjgdyhEm0uXLhCpRz1X7bpfELcxs1FqAGMLJFYGSOmUlrqrL4WROQT4FJ
+iZ0UFbqcujetJuWv9z+kBf9s/5r3ZoGd5VXNnNKQz7BV/
+wxjPk9Ui184ZHOOL2xM7BStdEDgTuMAbUsI5qDMCIE77g1JqZGWpeRuFf/
voJ2TNM7yUEikvf52UOihQbdiT6+feaK9urR4wcPDIwDB0/p8tObmVOCmKbhzVtXsnAjylpSk/
YHQ4fj6cxsMKSVz9IZihpga/DX12ki7Oacj/AJeGFIZYjBo/Kr2iEtoq5b0YQfC8Lp
+CItAelNzFzJ5IXbhmC8WtuhIk6ylQP5xyPofdlOJ1KBw46QckdxnMl3asO7rI4lHj2oXB4QiKVwgCbmvTw5U5O7NFB3rbp
UGTy7MHhf2hT9CbdJfxUUMXuv30nqc9xxkNxDozRw1KEAaG
+Ng/4p3WGmLRcSFj1JUHODkJyrubrD0djb5f4gNklKe4aNvsDF0tff4AUUQ0d/gYaLpdLr8/
sREhsfIzSgqKmyv9bi9QAAAAAAAAAAAAAAAAAAAAAAAAAABIgLTs="
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
```

```
}
```

Example例: 外部 mu データを使用して署名を検証する

このコマンドを使用して、ML-DSA パブリックキーを使用して外部 mu データの署名を検証します。

```
aws-cloudhsm > crypto verify ml-dsa \
    --key-filter attr.label=ml-dsa-public-key-example \
    --data-type external-mu \
    --hash-function shake256 \
    --data DgFbkWgb9fbejQUSbJ09/
Nmp7phb714RTkWdNAwQqd9u7M7XCWbI8EoaIeGcZCUwIzaG6kDavZDYxCcaEicbTw== \
    --signature "jxSKx9W/5Tu0lFySyrJPpNo4v/
tcNGfeC4Zt/0ubAAt7IQxH8MEjwcA7OO9AyaQntxDpzua8m+m5fEwiVrwmH
+zZhgw2EvXrFDU7zz6n6RRUI2RyXG6siistsBPwDZUU1G8oMp33k3E4LEL1al1AaYDncO9fQx1nS5wjg33OxzQbZi7GqRKB
+1MCPlSC3TshojRXlg2ELx32rRdzGhWAGZ/
A8qa9FIcS6LZlF35NZiJbK3W2mIFl8U5S2ZMXgIdfqm0wSNDibB4p23AR4SVz1Wghbu1To0pbeeaz74LxbZ3CIhpW
+UmtF8VZfGw/AbOQK2K1Z27VQuIc2Ru3L6PpSuDl4LM96ZdGPW6n/xjBMax83pAz9qwm
+cy/yjD28GCUKqwzNrjxB49ak6RGCYiUjCFG9UfEQ46xmJjOhvGaepf9XqjqmcO66rk
+mGThL96CGOpz1+QI91OQRSu3bDV9g2+Zl
+vEArvTUPLV5nbqZ598Uppswfg94n248S1P8M/5vkjiYBUD8asNcuQp2lQBXL4iDpuazewUyTLy2GADbwlfEYwxQ
+xELWxE4Gv8HMiQX2x97pC2BNjxlN1z4wMoPqsVlw
+G5xS2juTepUW3G9O0jgHvIVLuq2qC0XhSPRAvxWBE3iFVTDXwlpfaqjdRuhp5UwUfrL6m
+jKuLDRFxtzGFUmUfHtImSoZ0oNiQl+45Tlf+AZsYJno4chx5lQkVca7KI8jkvDLRqoxZFPb3T/
GlHoQ1R2y9EzB6LI0dCSAGWiU1kRDXcoXYls4Q6JfDZW6TRj0IAo0StVT7Au1S
+N+jPPKfpU7lJfROJglbt6gQM7rGy8naKsfEfBij+ufd89iKC+qwDP2WSx
+XsLMzfBBWHA8hJMGQHwEHeMvrjTJaSgKSP0M4pdcAV8g5A5TIaLgic4pszSVmc2PDmGw6ssAvS2M3SbnGCxUIcC7ziI74X
dy/uiZ8CirYhcvCH7jUsWR4J2JO+vAjnQJZZRXmUmmh3HHQoihV/P2kpmM8YxIVein1+
+gizJC9eijEuDh4rTr3SinO8vMoiclUG0+ah3yEKF6LjqtbWnNHY8Hnvm2w22yX1jjK4fsspsc9XbS3Og9un8qkaSpLphtz
f3CYiGLLFKupYjAKckU5J6t6LEdP8xoQF3alGVo/2lGQGIJrlg/qS2/7XpK5feeERtDxTCVZZhFj2w27OMsLTX/
Al0Koz8KCQFk5OWjQEfY3yicwP/+JB/ftQ
+5Uhp3CmOqSqNU7hRgL5e6QYi1E3D1KB8Je35Lz2PuzuClTttgWaExXe747fPvAbu3OJEjApdCYkaj0VokYhk2LH
+icpsCQfnMQjddei8WRyPMehp64PJRAXcCuX9qPiwfeCNxjplG+0PnY1SYKVRCQdbuuWORrAbLv8q7wSpLVB2A/
R0MISoZT6IGLiLxKbh7KZ7cSCV0rypD1+2Z7lYbvaUztP2CniJzqOuCbMDhlFzelbQaxY9UBlCiWjpxj7WShBnJuDD23Lhj
++W3mpaYBhUDMGMX0EWluSF7q2hAH8HcfMOUYDjaoV
+MoJupVb1yA6+NM5b6PmZR7RPiGPVBpoRR11GUAI9icFfYOr6gsc5yJZDsGtURz1PzpnSb21ZPlib9tHRW88kl
+2GijZes7DvyHhyavvfeO5M83uTCaYuohrVxKMJVewpW+O6nnPDxi
+dlWZ0NRQoU1CibRQiQi3ExDpmdQWMHHjkqVxxx6cnvn7wH60ztq5lsu3FoZ5gJSV8qabs3yjeh/
Tv5KlA7CaSVqcsYkqMEU2aT40Nysk2PO3v5b8ZmR9uU6YbQx7une+RZcmDhYXKtTG6NFaSNJjU3RwL805vaX/
hTVBc5o4jNK40JNIo1efrLzKp0V/cYk6/uBAoDAS4dbVhrf1GAyqlAbTc52dJJdJhzYaBJxddcdT
+CJAS8J1omVcPeMKn+Ny+Fy2sR53fHbP2E7OExbfYxToktPC4BRucIeFKWoYHYTSsYnennIbZBBfT4rPc7cA/
lWP94oxgjpCwbwQykLXeHh+MOgEDkk/Z+Gw/
xweTOWp6reiGUhERkXtniK2f3qjcRT03b9K6wNA7M08AKDSEtm3jT9KHhlJH9Tb1wOpPhUmj2MbIvbyG2yZknmiDIIwQmhM
PuG1yZMn9z+4tKFk6PYwbZON1GPPsVmhgKp+8AVIyLKVOoDNHX4io0u+Xk3+hAu5qU3kEJV29crhOCCJ8G2tBF
```

```
+y0Ho7S3ECAhGqFDxEKEQUFjXvU0gr8yGrqvgP1yC9ZxCu6TR5EESKtJc/
X2Acqyjg2Hx1IABe8ztmPT79FJyiAKDz5XsIfgBp0/
b1gBc3QV0VjYx4SvuYlVB0bFHqQmfs1bD2Ah9iCqPCe3qQli8q1T677shw07ijbyAYD6rfZqUDTwZr6EAQMC90v9sSaQY/5
FCquY2jZb5fTlDBk6TLYB8mMN2DJP+cDJ0FfhHOrt9/QIvrud2ZoldhAFPAFA841AYWb+BDVcZmLsIh
+RrFS0+vqyNyGN3TvOdxXxobFYuXA3QfcjcM6zMG+zcntK4b552QwKhVRzD9cvGB4AUM0ab0ivF
+8sfZVKtTYHepP/
FUiXk0QnMBpwydVrw0drz3dYRmvVHLSpJJemSVrT4hYmBrN0nMPWiiz9fANBHHC6F0TOwzeqmlrpXPXIiWOAp8ox/
fF9HcrULwkqJ5Bu6GVwXSCe19nFWVjt+8FPaa2B3mRaBgz71OLcpG4/LUD0rF0lUBvebl/
r8sEE5K2lhuak6Y/rg4/brTwVrFZdaYcmBUOSmboUpvox08HLY57XhJe8YoCQGQUezkL
+ZhXGMaYmphxmEp/4aWs6PvOfOWh849FCfQp9aEqfiX3d1rqNFK6NkLBSNEYf2D
+ikY/dLW98XUqDPHBipSlLY926ThO1Fc+BUFeeR9g6czyiB3K2Dcgi70cb61kAQ/
hHWNaaCdWfOdC7swjOlSlTMBo7CF6K
+FMA2uM9+8ZGOdlQOQUEgSutETmG9vNraGhC3qSZyhJBRtiDLSCxEQYu9BpgDDiAkK3ySpqu1urvD0uXs8fL5CiErOkBcZm
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: ファイルパスから署名を検証する

このコマンドを使用して、データと署名のファイルパスを使用して署名を検証します。

```
aws-cloudhsm > crypto verify ml-dsa \
    --key-filter attr.label=ml-dsa-public-key-example \
    --data-path /tmp/data.bin \
    --data-type raw \
    --hash-function shake256 \
    --signature-path /tmp/signature.bin
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

検証する Base64 エンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

検証するデータの場所を指定します。

必須: はい (データを通じて提供される場合を除く)

<HASH_FUNCTION>

ML-DSA で使用するハッシュ関数を指定します。

有効な値:

- シェイク256

必須: はい

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<SIGNATURE>

Base64 でエンコードされた署名。

必須: はい (署名パスを通じて提供される場合を除く)

<SIGNATURE_PATH>

署名の場所を指定します。

必須: はい (署名で提供される場合を除く)

<DATA_TYPE>

提供されたデータのデータ型を指定します。ハッシュされていないデータ`raw`には を使用し、計算された外部 mu データ`external-mu`には を使用します。

有効な値:

- raw
- external-mu

必須: はい

関連トピック

- CloudHSM CLI の暗号化検証カテゴリ
- CloudHSM CLI で ML-DSA メカニズムを使用して署名を生成する
- CloudHSM CLI を使用して非対称 ML-DSA キーペアを生成する

CloudHSM CLI で RSA-PKCS メカニズムを使用して署名された署名を検証する

CloudHSM CLI の crypto verify rsa-pkcs コマンドを使用して、次のオペレーションを完了します。

- 指定されたパブリックキーによって HSM でファイルが署名されていることを確認します。
- `RSA-PKCS` 署名メカニズムを使用して署名が生成されたことを検証します。
- 署名されたファイルをソースファイルと比較し、両者が指定された rsa パブリックキーと署名メカニズムに基づいて暗号的に関連するかどうかを判断します。

crypto verify rsa-pkcs コマンドを使用するには、まず AWS CloudHSM クラスターに RSA パブリックキーが必要です。

> **Note**
>
> CloudHSM CLI の CloudHSM CLI の暗号化署名カテゴリ サブコマンドを使用して署名を生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help crypto verify rsa-pkcs
Verify with the RSA-PKCS mechanism

Usage: crypto verify rsa-pkcs --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-
path <SIGNATURE_PATH>|--signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
      --data-type <DATA_TYPE>
          The type of data passed in, either raw or digest [possible values: raw,
 digest]
  -h, --help
          Print help
```

例

これらの例は、crypto verify rsa-pkcs を使用して RSA-PKCS 署名メカニズムと SHA256 ハッシュ関数を使用して生成された署名を検証する方法を示しています。このコマンドは HSM でパブリックキーを使用します。

Example例: Base64 でエンコードされた署名を Base64 でエンコードされたデータで検証する

```
aws-cloudhsm > crypto verify rsa-pkcs \
    --hash-function sha256 \
    --key-filter attr.label=rsa-public \
    --data YWJjMTIz \
    --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: データファイルを使用して署名ファイルを検証する

```
aws-cloudhsm > crypto verify rsa-pkcs \
    --hash-function sha256 \
    --key-filter attr.label=rsa-public \
    --data-path data.txt \
    --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: 偽の署名関係を証明する

このコマンドは、無効なデータが、ラベル rsa-public 付きのパブリックキーによって、/home/signature にある署名を生成する RSAPKCS 署名メカニズムを使用して署名されたかどうかを検

証します。指定の引数が真の署名関係を構成していないため、コマンドは、エラーメッセージを返します。

```
aws-cloudhsm > crypto verify rsa-pkcs \
    --hash-function sha256 \
    --key-filter attr.label=rsa-public \
    --data aW52YWxpZA== \
    --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データパスを通じて提供される場合を除く)

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1

- sha224
- sha256
- sha384
- sha512

必須: はい

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<SIGNATURE>

Base64 でエンコードされた署名。

必須: はい (署名パスを通じて提供される場合を除く)

<SIGNATURE_PATH>

署名の場所を指定します。

必須: はい (署名パスを通じて提供される場合を除く)

<DATA_TYPE>

データパラメータの値を、署名アルゴリズムの一部としてハッシュ化するかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

RSA-PKCS の場合、データは RFC 8017、セクション 9.2 で規定されている DER エンコード形式で渡す必要があります。

有効な値:

- raw
- ダイジェスト

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI で RSA-PKCS-PSS メカニズムを使用して署名された署名を検証する

CloudHSM CLI の crypto sign rsa-pkcs-pss コマンドを使用して、次のオペレーションを完了します。

- 指定されたパブリックキーによって HSM でファイルが署名されていることを確認します。
- RSA-PKCS-PSS 署名メカニズムを使用して署名が生成されたことを検証します。
- 署名されたファイルをソースファイルと比較し、両者が指定された rsa パブリックキーと署名メカニズムに基づいて暗号的に関連するかどうかを判断します。

crypto verify rsa-pkcs-pss コマンドを使用するには、まず AWS CloudHSM クラスターに RSA パブリックキーが必要です。RSA パブリックキーをインポートするには、CloudHSM CLI で PEM 形式キーをインポートする コマンドを使用するかunwrap、 `verify` 属性を に設定しますtrue。

> **Note**
>
> CloudHSM CLI の CloudHSM CLI の暗号化署名カテゴリ サブコマンドを使用して署名を生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help crypto verify rsa-pkcs-pss
```

```
Verify with the RSA-PKCS-PSS mechanism

Usage: crypto verify rsa-pkcs-pss --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --salt-length >SALT_LENGTH< <--data-
path <DATA_PATH>|--data <DATA> <--signature-path <SIGNATURE_PATH>|--
signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
      --data-type <DATA_TYPE>
          The type of data passed in, either raw or digest [possible values: raw,
 digest]
      --mgf <MGF>
          The mask generation function [possible values: mgf1-sha1, mgf1-sha224, mgf1-
sha256, mgf1-sha384, mgf1-sha512]
      --salt-length <SALT_LENGTH>
          The salt length
  -h, --help
          Print help
```

例

これらの例は、crypto verify rsa-pkcs-pss を使用して、RSA-PKCS-PSS 署名メカニズムと SHA256 ハッシュ関数を使用して生成された署名を検証する方法を示しています。このコマンドは HSM でパブリックキーを使用します。

Example例: Base64 でエンコードされた署名を Base64 でエンコードされたデータで検証する

```
aws-cloudhsm > crypto verify rsa-pkcs-pss \
    --key-filter attr.label=rsa-public \
    --hash-function sha256 \
    --data YWJjMTIz \
    --salt-length 10 \
    --mgf mgf1-sha256 \
    --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: データファイルを使用して署名ファイルを検証する

```
aws-cloudhsm > crypto verify rsa-pkcs-pss \
    --key-filter attr.label=rsa-public \
    --hash-function sha256 \
    --data-path data.txt \
    --salt-length 10 \
    --mgf mgf1-sha256 \
    --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example例: 偽の署名関係を証明する

このコマンドは、無効なデータが、ラベル `rsa-public` 付きのパブリックキーによって、`/home/signature` にある署名を生成する RSAPKCSPSS 署名メカニズムを使用して署名されたかどうかを検証します。指定の引数が真の署名関係を構成していないため、コマンドは、エラーメッセージを返します。

```
aws-cloudhsm > crypto verify rsa-pkcs-pss \
    --key-filter attr.label=rsa-public \
    --hash-function sha256 \
    --data aW52YWxpZA== \
    --salt-length 10 \
    --mgf mgf1-sha256 \
    --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<DATA>

署名対象の Base64 でエンコードされたデータ。

必須: はい (データパスを通じて提供される場合を除く)

<DATA_PATH>

署名するデータの場所を指定します。

必須: はい (データパスを通じて提供される場合を除く)

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1
- sha224

- sha256
- sha384
- sha512

必須: はい

<KEY_FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<MFG>

マスク生成関数を指定します。

> **Note**
>
> マスク生成関数のハッシュ関数は、署名メカニズムのハッシュ関数と一致する必要があります。

有効な値:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必須: はい

<SIGNATURE>

Base64 でエンコードされた署名。

必須: はい (署名パスを通じて提供される場合を除く)

<SIGNATURE_PATH>

署名の場所を指定します。

必須: はい (署名パスを通じて提供される場合を除く)

<DATA_TYPE>

データパラメータの値を、署名アルゴリズムの一部としてハッシュ化するかどうかを指定します。ハッシュされていないデータには `raw` を使用し、すでにハッシュされているダイジェストには `digest` を使用します。

有効な値:

- raw
- ダイジェスト

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI のキーカテゴリ

CloudHSM CLI では、key はコマンドのグループの親カテゴリであり、親カテゴリと組み合わせると、キーに固有のコマンドが作成されます。現在、このカテゴリは次のコマンドで構成されています。

- 削除
- generate-file
- key generate-asymmetric-pair
  - key generate-asymmetric-pair rsa
  - key generate-asymmetric-pair ec
  - キーgenerate-asymmetric-pair ml-dsa
- key generate-symmetric
  - key generate-symmetric aes

  - key generate-symmetric generic-secret
- import pem
- リスト
- レプリケーション
- セットの属性
- 使用
- unshare
- unwrap
- wrap

CloudHSM CLI でキーを削除する

CloudHSM CLI の key delete コマンドを使用して、 AWS CloudHSM クラスターからキーを削除します。一度に削除できるキーは 1 つだけです。キーペアの一方のキーを削除しても、ペアの他方のキーには影響がありません。

キーを作成し、そのキーを所有している CU のみがキーを削除できます。キーを共有しているが所有者ではないユーザーは、暗号化オペレーションでキーを使用できますが、削除することはできません。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key delete
Delete a key in the HSM cluster

Usage: key delete [OPTIONS] --filter [<FILTER>...]
```

```
Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --filter [<FILTER>...]     Key reference (e.g. key-reference=0xabc)
 or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for deletion
  -h, --help                     Print help
```

例

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを削除対象として選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

関連トピック

- CloudHSM CLI でユーザーのキーを一覧表示する

- CloudHSM CLI で非対称キーをエクスポートする
- CloudHSM CLI を使用してキーの共有を解除する
- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーをフィルタリングする

CloudHSM CLI で非対称キーをエクスポートする

CloudHSM CLI の key generate-file コマンドを使用して、ハードウェアセキュリティモジュール (HSM) から非対称キーをエクスポートします。ターゲットがプライベートキーの場合、プライベートキーへの参照はフェイク PEM 形式でエクスポートされます。ターゲットがパブリックキーの場合、パブリックキーバイトは PEM 形式でエクスポートされます。

フェイク PEM ファイルは、実際のプライベートキーマテリアルを含むわけではなく、HSM のプライベートキーを参照するため、ウェブサーバーから AWS CloudHSMへの SSL/TLS オフロードを確立するために使用できます。詳細については、「SSL/TLS オフロード」を参照してください。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key generate-file
Generate a key file from a key in the HSM cluster. This command does not export any
 private key data from the HSM

Usage: key generate-file --encoding <ENCODING> --path <PATH> --filter [<FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
      --encoding <ENCODING>
          Encoding format for the key file

          Possible values:
          - reference-pem:          PEM formatted key reference (supports private keys)
          - pem:                    PEM format (supports public keys)
          - cloudhsm-reference-pem: CloudHSM reference PEM format (supports private
 keys)

      --path <PATH>
          Filepath where the key file will be written

      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for file generation

  -h, --help
          Print help (see a summary with '-h')
```

例

これらの例は、 key generate-file を使用して AWS CloudHSM クラスター内のキーからキーファイルを生成する方法を示しています。

Example RSA または EC プライベートキーのリファレンス PEM ファイルを生成する

```
aws-cloudhsm > key generate-file --encoding reference-pem --path /tmp/ec-private-
key.pem --filter attr.label="ec-test-private-key"
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

Example Ed25519 または ML-DSA プライベートキーの CloudHSM リファレンス PEM ファイルを生成する

```
aws-cloudhsm > key generate-file --encoding cloudhsm-reference-pem --path /tmp/ed25519-
private-key.pem --filter attr.label="ed25519-test-private-key"
{
```

```
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを削除対象として選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: いいえ

<ENCODING>

キーファイルのエンコード形式を指定します

必須: はい

<PATH>

キーファイルが書き込まれるファイルパスを指定します

必須: はい

KSP キー参照を生成します (Windows)

> **Note**
>
> この機能は、SDK バージョン 5.16.0 以降でのみ使用できます。

前提条件

- KSP キー参照は Windows プラットフォームでのみ生成できます。
- Crypto User (CU) としてサインインする必要があります。

ファイルの場所

デフォルトでは、AWS CloudHSM は生成されたファイルを C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition に保存します。

別の場所を指定するには、--path パラメータを使用します。

構文

```
aws-cloudhsm > help key generate-file --encoding ksp-key-reference
Generate a key file from a key in the HSM cluster. This command does not export any
 private key data from the HSM

Usage: key generate-file --encoding <ENCODING> --path <PATH> --filter [<FILTER>...]

Options:
      --encoding <ENCODING>
        Encoding format for the key file

        Possible values:
        - reference-pem:     PEM formatted key reference (supports private keys)
        - pem:               PEM format (supports public keys)
        - ksp-key-reference: KSP key reference format

      --cluster-id <CLUSTER_ID>
        Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided with multiple clusters configured, will error

      --path <PATH>
        Directory path where the key file will be written

      --filter [<FILTER>...]
        Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for file generation

      --all
```

```
        Generate ksp key reference for all available key pairs in HSM

  -h, --help
        Print help (see a summary with '-h')
```

例 – プライベートキーの属性フィルターを使用して KSP キー参照を生成する

次の例では、特定のラベルを持つプライベートキーの KSP キー参照を生成します。

Example

```
aws-cloudhsm > key generate-file --encoding ksp-key-reference --path  --filter
 attr.label="ec-test-private-key"
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

例 – すべてのキーペアの KSP キー参照を生成する

次の例では、クラスター内のすべてのキーペアの KSP キー参照を生成します。

Example

```
aws-cloudhsm > key generate-file --encoding ksp-key-reference --all
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーをフィルタリングする
- CloudHSM CLI の generate-asymmetric-pair カテゴリ
- CloudHSM CLI の generate-symmetric カテゴリ

CloudHSM CLI の generate-asymmetric-pair カテゴリ

CloudHSM CLI では、key generate-asymmetric-pair はコマンドグループの親カテゴリで、親カテゴリと組み合わせると非対称キーペアを生成するコマンドを作成します。現在、このカテゴリは次のコマンドで構成されています。

- key generate-asymmetric-pair ec
- キーgenerate-asymmetric-pair ml-dsa
- key generate-asymmetric-pair rsa

CloudHSM CLI を使用して非対称 EC キーペアを生成する

CloudHSM CLI の key asymmetric-pair ec コマンドを使用して、 AWS CloudHSM クラスターに非対称楕円曲線 (EC) キーペアを生成します。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key generate-asymmetric-pair ec
Generate an Elliptic-Curve Cryptography (ECC) key pair

Usage: key generate-asymmetric-pair ec [OPTIONS] --public-label <PUBLIC_LABEL> --
private-label <PRIVATE_LABEL> --curve <CURVE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --public-label <PUBLIC_LABEL>
          Label for the public key
      --private-label <PRIVATE_LABEL>
```

```
          Label for the private key
     --session
          Creates a session key pair that exists only in the current session. The key
 cannot be recovered after the session ends
     --curve <CURVE>
          Elliptic curve used to generate the key pair [possible values: prime256v1,
 secp256r1, secp224r1, secp384r1, secp256k1, secp521r1, ed25519]
     --public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated EC public key
 in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
     --private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated EC private
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
     --share-crypto-users [<SHARE_CRYPTO_USERS>...]
          Space separated list of Crypto User usernames to share the EC private key
 with
     --manage-private-key-quorum-value <MANAGE_PRIVATE_KEY_QUORUM_VALUE>
          The quorum value for key management operations for the private key
     --use-private-key-quorum-value <USE_PRIVATE_KEY_QUORUM_VALUE>
          The quorum value for key usage operations for the private key
 -h, --help
          Print help
```

例

以下の例では、key generate-asymmetric-pair ec コマンドを使用して EC キーペアを作成する方法を示します。

Example例: EC キーペアの作成

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp224r1 \
    --public-label ec-public-key-example \
    --private-label ec-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
```

```
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-public-key-example",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
        "curve": "secp224r1"
      }
    },
"private_key": {
      "key-reference": "0x000000000012000c",
      "key-info": {
        "key-owners": [
          {
```

```
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
        "curve": "secp224r1"
      }
    }
  }
}
```

Example例: オプションの属性を持つ EC キーペア作成

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp224r1 \
    --public-label ec-public-key-example \
    --private-label ec-private-key-example \
    --public-attributes encrypt=true \
    --private-attributes decrypt=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000002806eb",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-public-key-example",
        "id": "",
        "check-value": "0xedef86",
        "class": "public-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
```

```
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
"0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
        "curve": "secp224r1"
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280c82",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
        "id": "",
        "check-value": "0xedef86",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
```

```
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
        "curve": "secp224r1"
      }
    }
  }
}
```

Example例: クォーラム値を持つ EC キーペア作成

クォーラム制御を使用してキーを生成する場合、キーには、キーの最大クォーラム値と同じ数以上のユーザーを関連付ける必要があります。関連付けられるユーザーには、キー所有者と、そのキーが共有されている Crypto User が含まれます。キーを共有すべき最小ユーザー数を決定するには、キー使用クォーラム値とキー管理クォーラム値のうち大きい方のクォーラム値を取得し、デフォルトでキーに関連付けられているキー所有者を考慮して 1 を差し引きます。キーをさらに多くのユーザーと共有するには、CloudHSM CLI を使用してキーを共有する コマンドを使用します。

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp224r1 \
    --public-label ec-public-key-example \
    --private-label ec-private-key-example \
    --public-attributes verify=true \
    --private-attributes sign=true
    --share-crypto-users cu2 cu3 cu4 \
    --manage-private-key-quorum-value 4 \
    --use-private-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000002806eb",
      "key-info": {
        "key-owners": [
```

```
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-public-key-example",
        "id": "",
        "check-value": "0xedef86",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
"0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
        "curve": "secp224r1"
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280c82",
```

```
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "cu2",
            "key-coverage": "full"
          },
          {
            "username": "cu3",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
        ],
        "key-quorum-values": {
          "manage-key-quorum-value": 4,
          "use-key-quorum-value": 2
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
        "id": "",
        "check-value": "0xedef86",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
```

```
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
        "curve": "secp224r1"
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<CURVE>

楕円曲線の識別子を指定します。

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1
- ed25519 (非 FIPS モードの hsm2m.medium インスタンスでのみサポート)

必須: はい

<PUBLIC_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された EC パブリックキーに設定するキー属性のスペース区切りリストを指定します (例: `verify=true`)。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<PUBLIC_LABEL>

パブリックキーのユーザー定義ラベルを指定します。label に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<PRIVATE_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された EC プライベートキーに設定するキー属性のスペース区切りリストを指定します (例: sign=true)。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<PRIVATE_LABEL>

プライベートキーのユーザー定義ラベルを指定します。label に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

デフォルトでは、生成されるキーは永続 (トークン) キーです。<SESSION> で渡すことでこれが変わり、この引数で生成されたキーがセッション (エフェメラル) キーであることが保証されます。

必須: いいえ

<SHARE_CRYPTO_USERS>

EC プライベートキーを共有する Crypto User ユーザー名のスペース区切りリストを指定します。

必須: いいえ

<MANAGE_PRIVATE_KEY_QUORUM_VALUE>

プライベートキーのキー管理オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

<USE_PRIVATE_KEY_QUORUM_VALUE>

プライベートキーのキー使用オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーをフィルタリングする

CloudHSM CLI を使用して非対称 ML-DSA キーペアを生成する

CloudHSM CLI の key generate-asymmetric-pair ml-dsa コマンドを使用して、 AWS CloudHSM クラスターに非対称 ML-DSA キーペアを生成します。ML-DSA はポスト量子デジタル署名アルゴリズムです。量子耐性アルゴリズムを使用してデータに署名または検証する必要がある場合は、ML-DSA キーペアを生成します。

> **Note**
>
> 2026 年 9 月 1 日以降、ML-DSA は hsm2m.medium クラスターの FIPS モードで使用できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key generate-asymmetric-pair ml-dsa
Generate an ML-DSA key pair

Usage: key generate-asymmetric-pair ml-dsa [OPTIONS] --public-label <PUBLIC_LABEL> --
private-label <PRIVATE_LABEL> --mldsa-algorithm <MLDSA_ALGORITHM>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --public-label <PUBLIC_LABEL>
          Label for the public key
      --private-label <PRIVATE_LABEL>
          Label for the private key
      --session
          Creates a session key pair that exists only in the current session. The key
 cannot be recovered after the session ends
      --mldsa-algorithm <MLDSA_ALGORITHM>
          ML-DSA algorithm [possible values: ML-DSA-44, ML-DSA-65, ML-DSA-87]
      --public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated ML-DSA public
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated ML-DSA
 private key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --share-crypto-users [<SHARE_CRYPTO_USERS>...]
          Space separated list of Crypto User usernames to share the ML-DSA key with
      --manage-private-key-quorum-value <MANAGE_PRIVATE_KEY_QUORUM_VALUE>
          The quorum value for key management operations for the private key
      --use-private-key-quorum-value <USE_PRIVATE_KEY_QUORUM_VALUE>
```

```
          The quorum value for key usage operations for the private key
  -h, --help
          Print help
```

例

これらの例は、 key generate-asymmetric-pair ml-dsaを使用して ML-DSA キーペアを作成する方法を示しています。

Example例: ML-DSA-44 キーペアを作成する

```
aws-cloudhsm > key generate-asymmetric-pair ml-dsa \
    --mldsa-algorithm ML-DSA-44 \
    --public-label ml-dsa-public-key-example \
    --private-label ml-dsa-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000000240b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ml-dsa",
        "label": "ml-dsa-public-key-example",
        "id": "0x",
        "check-value": "0x988721",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
```

```
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "mldsa-algorithm": "ML-DSA-44",
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1312
      }
    },
    "private_key": {
      "key-reference": "0x0000000000000f63",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ml-dsa",
        "label": "ml-dsa-private-key-example",
        "id": "0x",
        "check-value": "0x988721",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
```

```
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "mldsa-algorithm": "ML-DSA-44",
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 2560
      }
    }
  }
}
```

Example例: オプションの属性を使用して ML-DSA キーペアを作成する

```
aws-cloudhsm > key generate-asymmetric-pair ml-dsa \
    --mldsa-algorithm ML-DSA-65 \
    --public-label ml-dsa-public-key-example \
    --private-label ml-dsa-private-key-example \
    --public-attributes verify=true \
    --private-attributes sign=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000002806eb",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
```

```
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ml-dsa",
        "label": "ml-dsa-public-key-example",
        "mldsa-algorithm": "ML-DSA-65",
        "verify": true,
        "key-length-bytes": 1952
      }
    },
    "private_key": {
      "key-reference": "0x00000000002806ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ml-dsa",
        "label": "ml-dsa-private-key-example",
        "mldsa-algorithm": "ML-DSA-65",
        "sign": true,
        "key-length-bytes": 4032
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<MLDSA_ALGORITHM>

ML-DSA アルゴリズムを指定します。

- ML-DSA-44
- ML-DSA-65
- ML-DSA-87

必須: はい

<PUBLIC_KEY_ATTRIBUTES>

生成された ML-DSA パブリックキーに設定するキー属性のスペース区切りリストを の形式で指定します KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例: `verify=true`)。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<PUBLIC_LABEL>

パブリックキーのユーザー定義ラベルを指定します。`label` に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<PRIVATE_KEY_ATTRIBUTES>

生成された ML-DSA プライベートキーに設定するキー属性のスペース区切りリストを の形式で指定します KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例: `sign=true`)。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<PRIVATE_LABEL>

プライベートキーのユーザー定義ラベルを指定します。label に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、キーを短時間だけ必要とする場合に使用します。

デフォルトでは、HSM は永続的 (トークン) キーを生成します。<SESSION> を渡すと変更されるため、HSM は代わりにセッション (エフェメラル) キーを生成します。

必須: いいえ

<SHARE_CRYPTO_USERS>

ML-DSA プライベートキーを共有する Crypto ユーザーユーザー名のスペース区切りリストを指定します。

必須: いいえ

<MANAGE_PRIVATE_KEY_QUORUM_VALUE>

プライベートキーのキー管理オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、キーが共有されているユーザーとキー所有者が含まれます。最大値は 8 です。

必須: いいえ

<USE_PRIVATE_KEY_QUORUM_VALUE>

プライベートキーのキー使用オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、キーが共有されているユーザーとキー所有者が含まれます。最大値は 8 です。

必須: いいえ

関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI で ML-DSA メカニズムを使用して署名を生成する
- CloudHSM CLI で ML-DSA メカニズムで署名された署名を検証する

CloudHSM CLI で非対称 RSA キーペアを生成する

CloudHSM CLI の key generate-asymmetric-pair rsa コマンドを使用して、 AWS CloudHSM クラスターに非対称 RSA キーペアを生成します。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key generate-asymmetric-pair rsa
Generate an RSA key pair

Usage: key generate-asymmetric-pair rsa [OPTIONS] --public-label <PUBLIC_LABEL>
 --private-label <PRIVATE_LABEL> --modulus-size-bits <MODULUS_SIZE_BITS> --public-
exponent <PUBLIC_EXPONENT>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --public-label <PUBLIC_LABEL>
          Label for the public key
      --private-label <PRIVATE_LABEL>
          Label for the private key
      --session
          Creates a session key pair that exists only in the current session. The key
 cannot be recovered after the session ends
```

```
      --modulus-size-bits <MODULUS_SIZE_BITS>
          Modulus size in bits used to generate the RSA key pair
      --public-exponent <PUBLIC_EXPONENT>
          Public exponent used to generate the RSA key pair
      --public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated RSA public
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated RSA private
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --share-crypto-users [<SHARE_CRYPTO_USERS>...]
          Space separated list of Crypto User usernames to share the RSA key with
      --manage-private-key-quorum-value <MANAGE_PRIVATE_KEY_QUORUM_VALUE>
          The quorum value for key management operations for the private key
      --use-private-key-quorum-value <USE_PRIVATE_KEY_QUORUM_VALUE>
          The quorum value for key usage operations for the private key
  -h, --help
          Print help
```

例

以下の例では、key generate-asymmetric-pair rsa を使用して RSA キーペアを作成する方法を示します。

Example例: RSA キーペアの作成

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000160010",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
```

```
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x498e1f",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
e89a065e7d1a46ced96b46b909db2ab6be871ee700fd0a448b6e975bb64cae77c49008749212463e37a577baa57ce3e
bcebb7d20bd6df1948ae336ae23b52d73b7f3b6acc2543edb6358e08d326d280ce489571f4d34e316a2ea1904d513ca
        "modulus-size-bits": 2048
      }
    },
"private_key": {
      "key-reference": "0x0000000000160011",
      "key-info": {
        "key-owners": [
```

```
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "",
        "check-value": "0x498e1f",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
"0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
        "modulus-size-bits": 2048
      }
    }
  }
```

```
}
```

Example例: オプションの属性を含む RSA キーペアの作成

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes encrypt=true \
--private-attributes decrypt=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000280cc8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "public-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
```

```
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa01
f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d510
133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e834697e
ac3160f0ca9725d38318b7",
        "modulus-size-bits": 2048
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280cc7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "private-key",
```

```
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
        "modulus-size-bits": 2048
      }
    }
  }
}
```

Example例: クォーラム値を持つ RSA キーペア作成

クォーラム制御を使用してキーを生成する場合、キーには、キーの最大クォーラム値と同じ数以上のユーザーを関連付ける必要があります。関連付けられるユーザーには、キー所有者と、そのキーが共有されている Crypto User が含まれます。キーを共有すべき最小ユーザー数を決定するには、キー使用クォーラム値とキー管理クォーラム値のうち大きい方のクォーラム値を取得し、デフォルトでキーに関連付けられているキー所有者を考慮して 1 を差し引きます。キーをさらに多くのユーザーと共有するには、CloudHSM CLI を使用してキーを共有する コマンドを使用します。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes verify=true \
```

```
--private-attributes sign=true
--share-crypto-users cu2 cu3 cu4 \
--manage-private-key-quorum-value 4 \
--use-private-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000280cc8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": true,
        "trusted": false,
        "unwrap": false,
```

```
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa01
f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d510
133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e834697e
ac3160f0ca9725d38318b7",
        "modulus-size-bits": 2048
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280cc7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "cu2",
            "key-coverage": "full"
          },
          {
            "username": "cu3",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
        ],
        "key-quorum-values": {
          "manage-key-quorum-value": 4,
          "use-key-quorum-value": 2
        },
        "cluster-coverage": "full"
      },
      "attributes": {
```

```
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
        "modulus-size-bits": 2048
      }
    }
  }
}
```


引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<MODULUS_SIZE_BITS>

モジュラスの長さをビット単位で指定します。最小値は 2,048 です。

必須: はい

<PRIVATE_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された RSA プライベートキーに設定するキー属性のスペース区切りリストを指定します (例: `sign=true`)。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<PRIVATE_LABEL>

プライベートキーのユーザー定義ラベルを指定します。`label` に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<PUBLIC_EXPONENT>

パブリック指数を指定します。値は、65537 以上の奇数にする必要があります

必須: はい

<PUBLIC_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された RSA パブリックキーに設定するキー属性のスペース区切りリストを指定します (例: `verify=true`)

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<PUBLIC_LABEL>

パブリックキーのユーザー定義ラベルを指定します。`label` に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

デフォルトでは、生成されるキーは永続 (トークン) キーです。<SESSION> で渡すことでこれが変わり、この引数で生成されたキーがセッション (エフェメラル) キーであることが保証されます。

必須: いいえ

<SHARE_CRYPTO_USERS>

RSAEC プライベートキーを共有する Crypto User ユーザー名のスペース区切りリストを指定します。

必須: いいえ

<MANAGE_PRIVATE_KEY_QUORUM_VALUE>

プライベートキーのキー管理オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

<USE_PRIVATE_KEY_QUORUM_VALUE>

プライベートキーのキー使用オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーをフィルタリングする

CloudHSM CLI の generate-symmetric カテゴリ

CloudHSM CLI では、key generate-symmetric はコマンドグループの親カテゴリであり、親カテゴリと組み合わせると対称キーを生成するコマンドを作成します。現在、このカテゴリは次のコマンドで構成されています。

- key generate-symmetric aes
- key generate-symmetric generic-secret

CloudHSM CLI で対称 AES キーを生成する

CloudHSM CLI の key generate-symmetric aes コマンドを使用して、 AWS CloudHSM クラスターに対称 AES キーを生成します。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key generate-symmetric aes
Generate an AES key

Usage: key generate-symmetric aes [OPTIONS] --label <LABEL> --key-length-
bytes <KEY_LENGTH_BYTES>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --label <LABEL>
          Label for the key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
```

```
      --key-length-bytes <KEY_LENGTH_BYTES>
          Key length in bytes
      --attributes [<KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated AES key in
 the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --share-crypto-users [<SHARE_CRYPTO_USERS>...]
          Space separated list of Crypto User usernames to share the AES key with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>
          The quorum value for key management operations
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE>
          The quorum value for key usage operations
  -h, --help
          Print help
```

例

以下の例では、key generate-symmetric aes コマンドを使って AES キーを作成する方法を示します。

Example例: AES キーの作成

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
```

```
      "attributes": {
        "key-type": "aes",
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
      }
    }
  }
}
```

Example例: オプションの属性を含む AES キーペアの作成

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24 \
--attributes decrypt=true encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
```

```
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
      }
    }
  }
}
```

Example例: クォーラム値を含む AES キーの作成

クォーラム制御を使用してキーを生成する場合、キーには、キーの最大クォーラム値と同じ数以上のユーザーを関連付ける必要があります。関連付けられるユーザーには、キー所有者と、そのキーが共有されている Crypto User が含まれます。キーを共有すべき最小ユーザー数を決定するには、キー使用クォーラム値とキー管理クォーラム値のうち大きい方のクォーラム値を取得し、デフォルトでキーに関連付けられているキー所有者を考慮して 1 を差し引きます。キーをさらに多くのユーザーと共有するには、CloudHSM CLI を使用してキーを共有する コマンドを使用します。

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24 \
--attributes decrypt=true encrypt=true
--share-crypto-users cu2 cu3 cu4 \
--manage-key-quorum-value 4 \
--use-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "cu2",
            "key-coverage": "full"
          },
          {
            "username": "cu3",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
        ],
        "key-quorum-values": {
```

```
          "manage-key-quorum-value": 4,
          "use-key-quorum-value": 2
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された AES キーに設定するキー属性のスペース区切りリストを指定します (例: `sign=true`)。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<KEY-LENGTH-BYTES>

キーの長さをバイト単位で指定します。

有効な値:

- 16、24、32

必須: はい

<LABEL>

AES キーのユーザー定義ラベルを指定します。`label` に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

デフォルトでは、生成されるキーは永続 (トークン) キーです。<SESSION> で渡すことでこれが変わり、この引数で生成されたキーがセッション (エフェメラル) キーであることが保証されます。

必須: いいえ

<SHARE_CRYPTO_USERS>

AES キーを共有する Crypto User ユーザー名のスペース区切りリストを指定します。

必須: いいえ

<MANAGE_KEY_QUORUM_VALUE>

キー管理オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

<USE_KEY_QUORUM_VALUE>

キー使用オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーをフィルタリングする

CloudHSM CLI で対称汎用シークレットキーを生成する

CloudHSM CLI の key generate-symmetric generic-secret コマンドを使用して、 AWS CloudHSM クラスターに対称汎用シークレットキーを生成します。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > key help generate-symmetric generic-secret
Generate a generic secret key
```

```
Usage: key generate-symmetric generic-secret [OPTIONS] --label <LABEL> --key-length-
bytes <KEY_LENGTH_BYTES>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --label <LABEL>
          Label for the key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --key-length-bytes <KEY_LENGTH_BYTES>
          Key length in bytes
      --attributes [<KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated generic
 secret key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --share-crypto-users [<SHARE_CRYPTO_USERS>...]
          Space separated list of Crypto User usernames to share the generic secret key
 with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>
          The quorum value for key management operations
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE>
          The quorum value for key usage operations
  -h, --help
          Print help
```

例

次の例は、key generate-symmetric generic-secret コマンドを使って汎用シークレットキーを作成する方法を示しています。

Example例: 汎用シークレットキー作成

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
```

```
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 256
      }
    }
  }
}
```

Example例: オプションの属性を持つ汎用シークレットキー作成

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256 \
--attributes encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
```

```
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 256
      }
    }
  }
}
```

Example例: クォーラム値を持つ汎用シークレットキー作成

クォーラム制御を使用してキーを生成する場合、キーには、キーの最大クォーラム値と同じ数以上のユーザーを関連付ける必要があります。関連付けられるユーザーには、キー所有者と、そのキーが共有されている Crypto User が含まれます。キーを共有すべき最小ユーザー数を決定するには、キー使用クォーラム値とキー管理クォーラム値のうち大きい方のクォーラム値を取得し、デフォルトでキーに関連付けられているキー所有者を考慮して 1 を差し引きます。キーをさらに多くのユーザーと共有するには、CloudHSM CLI を使用してキーを共有する コマンドを使用します。

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256 \
--attributes encrypt=true
--share-crypto-users cu2 cu3 cu4 \
--manage-key-quorum-value 4 \
--use-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "cu2",
            "key-coverage": "full"
          },
```

```
          {
            "username": "cu3",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
        ],
        "key-quorum-values": {
          "manage-key-quorum-value": 4,
          "use-key-quorum-value": 2
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 256
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式で生成された AES キーに設定するキー属性のスペース区切りリストを指定します (例: `sign=true`)。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

<KEY-LENGTH-BYTES>

キーの長さをバイト単位で指定します。

有効な値:

- 1 ~ 800

必須: はい

<LABEL>

汎用シークレットキーのユーザー定義ラベルを指定します。`label` に許可される最大サイズは、クライアント SDK 5.11 以降では 127 文字です。クライアント SDK 5.10 以前では、制限は 126 文字です。

必須: はい

<SESSION>

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

デフォルトでは、生成されるキーは永続 (トークン) キーです。<SESSION> で渡すことでこれが変わり、この引数で生成されたキーがセッション (エフェメラル) キーであることが保証されます。

必須: いいえ

<SHARE_CRYPTO_USERS>

汎用シークレットキーを共有する Crypto User ユーザー名のスペース区切りリスト。

必須: いいえ

<MANAGE_KEY_QUORUM_VALUE>

キー管理オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

<USE_KEY_QUORUM_VALUE>

キー使用オペレーションのクォーラム値。この値は、キーが関連付けられているユーザーの数以下である必要があります。これには、そのキーが共有されているユーザーと、キー所有者が含まれます。最大値は 8 です。

必須: いいえ

関連トピック

- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーをフィルタリングする

CloudHSM CLI で PEM 形式キーをインポートする

の key import pem コマンドを使用して AWS CloudHSM 、PEM 形式キーをハードウェアセキュリティモジュール (HSM) にインポートします。このコマンドを使用すると、HSM の外部で生成されたパブリックキーをインポートできます。

> **Note**
>
> ML-DSA キーは、 key import pem コマンドを使用してインポートすることはできません。ML-DSA キーを使用するには、key generate-asymmetric-pair ml-dsa を使用して HSM で直接生成します。

> **Note**
>
> CloudHSM CLI で非対称キーをエクスポートする コマンドを使用して、パブリックキーから標準 PEM ファイルを作成するか、プライベートキーから参照 PEM ファイルを作成します。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key import pem
Import key from a PEM file

Usage: key import pem [OPTIONS] --path <PATH> --label <LABEL> --key-type-
class <KEY_TYPE_CLASS>
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --path <PATH>
          Path where the key is located in PEM format
      --label <LABEL>
          Label for the imported key
      --key-type-class <KEY_TYPE_CLASS>
```

```
          Key type and class of the imported key [possible values: ec-public, rsa-
public]
      --attributes [<IMPORT_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the imported key
  -h, --help
          Print help
```

例

この例では、key import pem コマンドを使用して、PEM 形式のファイルから RSA パブリックキーをインポートする方法を示しています。

Example例: RSA パブリックキーをインポートする

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key --key-
type-class rsa-public
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
```

```
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "modulus-size-bits": 2048
      }
    },
    "message": "Successfully imported key"
  }
}
```

Example例: オプションの属性を含む RSA パブリックキーをインポートする

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key-with-
attributes --key-type-class rsa-public --attributes verify=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
```

```
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key-with-attributes",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "modulus-size-bits": 2048
      }
    },
    "message": "Successfully imported key"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PATH>

キーファイルがあるファイルパスを指定します。

必須: はい

<LABEL>

インポートされたキーのユーザー定義ラベルを指定します。`label` の最大長は 126 文字です。

必須: はい

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス。

使用できる値:

- ec-public
- rsa-public

必須: はい

<IMPORT_KEY_ATTRIBUTES>

`KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式 (例: `sign=true`) でインポートされたキーに設定するキー属性のスペース区切りリストを指定します。サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: いいえ

関連トピック

- CloudHSM CLI の暗号化署名カテゴリ
- CloudHSM CLI の暗号化検証カテゴリ

CloudHSM CLI でユーザーのキーを一覧表示する

CloudHSM CLI の key list コマンドを使用して、 AWS CloudHSM クラスター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。出力には、そのユーザーが所有および共有しているキーと、CloudHSM クラスターのすべてのパブリックキーが含まれます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- 管理者 (CO)
- Crypto User (CU)

構文

```
aws-cloudhsm > help key list
List the keys the current user owns, shares, and all public keys in the HSM cluster

Usage: key list [OPTIONS]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select
 matching key(s) to list
      --max-items <MAX_ITEMS>
          The total number of items to return in the command's output. If the total
 number of items available is more than the value specified, a next-token is provided
 in the command's output. To resume pagination, provide the next-token value in the
 starting-token argument of a subsequent command [default: 10]
      --starting-token <STARTING_TOKEN>
          A token to specify where to start paginating. This is the next-token from a
 previously truncated response
  -v, --verbose
          If included, prints all attributes and key information for each matched key.
 By default each matched key only displays its key-reference and label attribute. This
 flag when used by Admins has no effect
  -h, --help
```

```
          Print help
```

例

次の例は、key list コマンドを実行するためのさまざまな方法を示しています。次の例は、Crypto User としての出力を示しています。

Example例: すべてのキーを検索 - デフォルト

このコマンドは、 AWS CloudHSM クラスターに存在するログインしているユーザーのキーを一覧表示します。

> **Note**
>
> 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。適切なページ分割オプションを使用して、出力として表示するキーの数を増やしたり減らしたりします。

```
aws-cloudhsm > key list
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000003d5",
        "attributes": {
          "label": "test_label_1"
        }
      },
      {
        "key-reference": "0x0000000000000626",
        "attributes": {
          "label": "test_label_2"
        }
      },.
      ...8 keys later...
    ],
    "total_key_count": 56,
    "returned_key_count": 10,
    "next_token": "10"
  }
```

```
}
```

Example例: すべてのキーを検索 - 詳細度

出力には、そのユーザーが所有および共有しているキーと、HSM のすべてのパブリックキーが含まれます。

> **Note**
>
> 注: 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 個だけ表示されます。適切なページ分割オプションを使用して、出力として表示するキーの数を増やしたり減らしたりします。

```
aws-cloudhsm > key list --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000012000c",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-private-key",
          "id": "",
          "check-value": "0x2a737d",
          "class": "private-key",
          "encrypt": false,
```

```
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": false,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 122,
          "ec-point":
"0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
          "curve": "secp224r1"
        }
      },
      {
        "key-reference": "0x000000000012000d",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-public-key",
          "id": "",
          "check-value": "0x2a737d",
```

```
          "class": "public-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": false,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": false,
          "sign": false,
          "trusted": false,
          "unwrap": false,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 57,
          "ec-point":
 "0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
          "curve": "secp224r1"
        }
      }
    ],
      ...8 keys later...
    "total_key_count": 1580,
    "returned_key_count": 10
  }
}
```

Example例: ページ分割されたリターン

次の例では、2 つのキーのみを表示するページ分割されたキーのサブセットを表示しています。次に、この例では次の 2 つのキーを表示する後続の呼び出しを行います。

```
aws-cloudhsm > key list --verbose --max-items 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
```

```
        "key-reference": "0x0000000000000030",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "98a6688d1d964ed7b45b9cec5c4b1909",
          "id": "",
          "check-value": "0xb28a46",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      },
      {
        "key-reference": "0x0000000000000042",
```

```
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "4ad6cdcbc02044e09fa954143efde233",
          "id": "",
          "check-value": "0xc98104",
          "class": "secret-key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": true,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 16
        }
      }
    ],
    "total_key_count": 1580,
    "returned_key_count": 2,
```

```
    "next_token": "2"
  }
}
```

次の 2 つのキーを表示するには、後続の呼び出しを行います。

```
aws-cloudhsm > key list --verbose --max-items 2 --starting-token 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000000081",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "6793b8439d044046982e5b895791e47f",
          "id": "",
          "check-value": "0x3f986f",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
```

```
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      },
      {
        "key-reference": "0x0000000000000089",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "56b30fa05c6741faab8f606d3b7fe105",
          "id": "",
          "check-value": "0xe9201a",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
```

```
            "sensitive": true,
            "sign": true,
            "trusted": false,
            "unwrap": false,
            "verify": true,
            "wrap": false,
            "wrap-with-trusted": false,
            "key-length-bytes": 32
          }
        }
      ],
      "total_key_count": 1580,
      "returned_key_count": 2,
      "next_token": "4"
    }
}
```

CloudHSM CLI でキーフィルタリングメカニズムがどのように機能するかを示すその他の例については、CloudHSM CLI を使用してキーをフィルタリングする を参照してください。

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキー (複数可) を一覧表示対象として選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: いいえ

<MAX_ITEMS>

コマンドの出力で返される項目の総数。使用可能な項目の総数が指定された値を上回る場合、コマンドの出力で next-token が提供されます。ページ分割を再開するには、後続コマンドの starting-token 引数で next-token 値を指定します。

必須: いいえ

<STARTING_TOKEN>

ページ分割を始める場所を指定するトークン。これは、以前に切り詰められたレスポンスからの next-token です。

必須: いいえ

<VERBOSE>

含まれている場合は、一致した各キーのすべての属性とキー情報を出力します。デフォルトでは、一致した各キーには key-reference とラベル属性のみが表示されます。管理者が使用する場合、このフラグは効果がありません。

必須: いいえ

関連トピック

- CloudHSM CLI でキーを削除する
- CloudHSM CLI で非対称キーをエクスポートする
- CloudHSM CLI を使用してキーの共有を解除する
- CloudHSM CLI のキー属性
- CloudHSM CLI を使用してキーをフィルタリングする

CloudHSM CLI でキーをレプリケートする

CloudHSM CLI の key replicate コマンドを使用して、ソース AWS CloudHSM クラスターから宛先 AWS CloudHSM クラスターにキーをレプリケートします。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- 管理者 (CO)
- Crypto User (CU)

> **Note**
>
> Crypto User は、このコマンドを使用するにはキーを所有している必要があります。

要件

- 送信元クラスターと送信先クラスターはクローンである必要があります。つまり、一方が他方のバックアップから作成されたか、どちらも共通のバックアップから作成されたということです。詳細については「バックアップからクラスターを作成する」を参照してください。
- キーのオーナーは、送信先クラスターに存在する必要があります。さらに、キーを共有しているユーザーがいる場合、それらのユーザーも送信先クラスターに存在する必要があります。
- このコマンドを実行するには、送信元クラスターと送信先クラスターの両方で、Crypto User または管理者としてログインしている必要があります。
  - 単一コマンドモードでは、コマンドは CLOUDHSM_PIN および CLOUDHSM_ROLE 環境変数を使用してソースクラスターで認証します。詳細については「シングルコマンドモード」を参照してください。送信先クラスターの認証情報を提供するには、DESTINATION_CLOUDHSM_PIN と DESTINATION_CLOUDHSM_ROLE の 2 つの追加の環境変数を設定する必要があります。

```
$ export DESTINATION_CLOUDHSM_ROLE=<role>
```

```
$ export DESTINATION_CLOUDHSM_PIN=<username:password>
```

  - インタラクティブモードでは、ユーザーは送信元クラスターと送信先クラスターの両方に明示的にログインする必要があります。

構文

```
aws-cloudhsm > help key replicate
Replicate a key from a source to a destination cluster

Usage: key replicate --filter [<FILTER>...] --source-cluster-id <SOURCE_CLUSTER_ID> --
destination-cluster-id <DESTINATION_CLUSTER_ID>

Options:
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select
 matching key on the source cluster
      --source-cluster-id <SOURCE_CLUSTER_ID>
          Source cluster ID
      --destination-cluster-id <DESTINATION_CLUSTER_ID>
          Destination cluster ID
  -h, --help
```

```
          Print help
```

例

Example例: キーをレプリケートする

このコマンドは、送信元のクラスターから送信先クラスター (クローン) にキーをレプリケートします。以下の例は、両方のクラスターに Crypto User としてログインしたときの出力を示しています。

```
crypto-user-1@cluster-1234abcdefg > key replicate \
      --filter attr.label=example-key \
      --source-cluster-id cluster-1234abcdefg \
      --destination-cluster-id cluster-2345bcdefgh
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x0000000000300006",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto-user-1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-key",
        "id": "0x",
        "check-value": "0x5e118e",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
```

```
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": true,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    },
    "message": "Successfully replicated key"
  }
}
```

引数

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを送信元クラスターで選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: はい

<SOURCE_CLUSTER_ID>

送信元クラスターの ID。

必須: はい

<DESTINATION_CLUSTER_ID>

送信先クラスターの ID。

必須: はい

関連トピック

- CloudHSM CLI を使用した複数のクラスターへの接続

CloudHSM CLI でキーの属性を設定する

CloudHSM CLI の key set-attribute コマンドを使用して、 AWS CloudHSM クラスター内のキーの属性を設定します。キーの属性を変更できるのは、キーを作成し、その結果キーを所有する CU のみです。

CloudHSM CLI で使用できる主な属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- このコマンドを実行できるのは Crypto User (CU) です。
- 管理者は信頼できる属性を設定できます。

要件

このコマンドを実行するには、CU としてログインする必要があります。信頼できる属性を設定するには、管理者ユーザーとしてログインする必要があります。

Syntax

```
aws-cloudhsm > help key set-attribute
Set an attribute for a key in the HSM cluster

Usage: cloudhsm-cli key set-attribute [OPTIONS] --filter [<FILTER>...] --
name <KEY_ATTRIBUTE> --value <KEY_ATTRIBUTE_VALUE>

Options:
      --cluster-id <CLUSTER_ID>          Unique Id to choose which of the clusters in
 the config file to run the operation against. If not provided, will fall back to the
 value provided when interactive mode was started, or error
      --filter [<FILTER>...]             Key reference (e.g. key-
reference=0xabc) or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key to modify
```

```
      --name <KEY_ATTRIBUTE>             Name of attribute to be set
      --value <KEY_ATTRIBUTE_VALUE>...   Attribute value to be set
      --approval <APPROVAL>             Filepath of signed quorum token file to approve
operation
 -h, --help                              Print help
```

例: キー属性の設定

次の例は、key set-attribute コマンドを使用してラベルを設定する方法を示しています。

Example

1. 次に示すように、ラベル my_key の付いたキーを使用してください。

```
aws-cloudhsm > key set-attribute --filter attr.label=my_key --name encrypt --value
 false
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

2. key list コマンドを使用して、encrypt 属性が変更されたことを確認します。

```
aws-cloudhsm > key list --filter attr.label=my_key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000006400ec",
        "key-info": {
          "key-owners": [
            {
              "username": "bob",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
```

```
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "my_key",
          "id": "",
          "check-value": "0x6bd9f7",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": true,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": true,
          "unwrap": true,
          "verify": true,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<KEY_ATTRIBUTE>

キーの属性の名前を指定します。

必須: はい

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを削除対象として選択します。

サポートされている CloudHSM CLI キー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください

必須: いいえ

<KEY_ATTRIBUTE_VALUE>

キーの属性の値を指定します。

必須: はい

<KEY_REFERENCE>

キーの 16 進数または 10 進数表現。 (キーハンドルなど)。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI を使用してキーをフィルタリングする
- CloudHSM CLI のキー属性

CloudHSM CLI を使用してキーを共有する

CloudHSM CLI の key share コマンドを使用して、 AWS CloudHSM クラスター内の他の CUs とキーを共有します。

キーを共有できるのは、キーを作成し、その結果キーを所有する CU のみです。キーを共有するユーザーは、暗号化オペレーションでキーを使用できますが、キーを削除、エクスポート、共有、共有解除、取得、またはラップすることはできません。さらに、これらのユーザーは キー属性 を変更できません。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key share
Share a key in the HSM cluster with another user

Usage: key share --filter [<FILTER>...] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for sharing

      --username <USERNAME>
          A username with which the key will be shared

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts
```

```
    --approval <APPROVAL>
        Filepath of signed quorum token file to approve operation

-h, --help
        Print help (see a summary with '-h')
```

例: キーを別の CU と共有する

以下の例は、key share コマンドを使用して CU `alice` とキーを共有する方法を示します。

Example

1. key share コマンドを実行して `alice` とキーを共有します。

```
aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}
```

2. key list コマンドを実行します。

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
```

```
            "key-coverage": "full"
          },
          {
            "username": "cu1",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
          {
            "username": "cu5",
            "key-coverage": "full"
          },
          {
            "username": "cu6",
            "key-coverage": "full"
          },
          {
            "username": "cu7",
            "key-coverage": "full"
          },
          {
            "username": "alice",
            "key-coverage": "full"
          }
        ],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa_key_to_share",
        "id": "",
        "check-value": "0xae8ff0",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
```

```
            "destroyable": true,
            "extractable": true,
            "local": true,
            "modifiable": true,
            "never-extractable": false,
            "private": true,
            "sensitive": true,
            "sign": true,
            "trusted": false,
            "unwrap": true,
            "verify": false,
            "wrap": false,
            "wrap-with-trusted": false,
            "key-length-bytes": 1219,
            "public-exponent": "0x010001",
            "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
            "modulus-size-bits": 2048
          }
        }
      ],
      "total_key_count": 1,
      "returned_key_count": 1
    }
}
```


3. 上記のリストで、alice が shared-users のリストに含まれている事を検証します

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: key-reference=0xabc) または attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のスペース区切りリスト。これに一致するキーを削除対象として選択します。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: はい

<USERNAME>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。このコマンドではユーザー名の大文字と小文字は区別されません。ユーザー名は常に小文字で表示されます。

必須: はい

<ROLE>

このユーザーに割り当てられるロールを指定します。このパラメータは必須です。ユーザーのロールを取得するには、ユーザーリストコマンドを使用します。HSM のユーザータイプの詳細については、「CloudHSM CLI の HSM ユーザータイプ」を参照してください。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI を使用してキーをフィルタリングする
- CloudHSM CLI のキー属性

CloudHSM CLI を使用してキーの共有を解除する

CloudHSM CLI の key unshare コマンドを使用して、 AWS CloudHSM クラスター内の他の CUs とキーの共有を解除します。

キーを共有解除できるのは、キーを作成し、その結果キーを所有する CU のみです。キーを共有しているユーザーは、暗号化オペレーションでキーを使用できますが、キーを削除、エクスポート、共有、または共有解除することはできません。さらに、これらのユーザーは キー属性 を変更できません。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key unshare
Unshare a key in the HSM cluster with another user

Usage: key unshare --filter [<FILTER>...] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for unsharing

      --username <USERNAME>
          A username with which the key will be unshared

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

  -h, --help
          Print help (see a summary with '-h')
```

例: 別の CU とのキーの共有を解除する

以下の例は、key unshare コマンドを使用して CU `alice` とのキーの共有を解除する方法を示しています。

Example

1. key list コマンドを実行し、alice と共有を解除したい特定のキーでフィルタリングします。

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
```

```
            },
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
```

```
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

2. shared-users 出力に alice が含まれていることを確認し、次の key unshare コマンドを実行して alice とのキーの共有を解除します。

```
aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
 attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key unshared successfully"
  }
}
```

3. key list コマンドをもう一度実行して、alice とキーが共有解除されたことを確認します。

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
```

```
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
```

```
            "verify": false,
            "wrap": false,
            "wrap-with-trusted": false,
            "key-length-bytes": 1219,
            "public-exponent": "0x010001",
            "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
            "modulus-size-bits": 2048
          }
        }
      ],
      "total_key_count": 1,
      "returned_key_count": 1
    }
}
```


引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。これに一致するキーを削除対象として選択します。

サポートされているキー属性のリストについては、「CloudHSM CLI のキー属性」を参照してください。

必須: はい

<USERNAME>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。このコマンドではユーザー名の大文字と小文字は区別されません。ユーザー名は常に小文字で表示されます。

必須: はい

<ROLE>

このユーザーに割り当てられるロールを指定します。このパラメータは必須です。ユーザーのロールを取得するには、ユーザーリストコマンドを使用します。HSM のユーザータイプの詳細については、「CloudHSM CLI の HSM ユーザータイプ」を参照してください。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI を使用してキーをフィルタリングする
- CloudHSM CLI のキー属性

CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI の key unwrap 親コマンドは、暗号化された (ラップされた) 対称または非対称プライベートキーをファイルから HSM 内にインポートします。このコマンドは、CloudHSM CLI のキーラップコマンド コマンドでラップされた暗号化されたキーをインポートするように設計されていますが、他のツールでラップされたキーをアンラップするためにも使用できます。ただし、このような場合は、PKCS#11 または JCE ソフトウェアライブラリを使用して、キーをラップ解除することをお勧めします。

> **Note**
>
> ML-DSA キーはラップ解除できません。ML-DSA キーを使用するには、key generate-asymmetric-pair ml-dsa を使用して HSM で直接生成します。

- aes-gcm
- aes-no-pad
- aes-pkcs5-pad
- aes-zero-pad

- cloudhsm-aes-gcm
- rsa-aes
- rsa-oaep
- rsa-pkcs

CloudHSM CLI を使用して AES-GCM でキーのラッピングを解除する

CloudHSM CLI の key unwrap aes-gcm コマンドを使用して、AES ラッピングキーと AES-GCM ラップ解除メカニズムを使用してペイロードキーのラップを解除してクラスターに入れます。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、local 属性は false に設定されます。

key unwrap aes-gcm コマンドを使用するには、 AWS CloudHSM クラスターに AES ラッピングキーがあり、そのunwrap属性を に設定する必要がありますtrue。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key unwrap aes-gcm
Usage: key unwrap aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-
bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> --iv <IV> <--
data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
```

```
          Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --share-crypto-users [<SHARE_CRYPTO_USERS;...]
          Space separated list of Crypto User usernames to share the unwrapped key with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
          The quorum value for key management operations for the unwrapped key
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
          The quorum value for key usage operations for the unwrapped key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
      --iv <IV>
          Initial value used to wrap the key, in hex
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
 -h, --help
          Print help
```

例

これらの例は、unwrap 属性値を true に設定した AES キーを使用して key unwrap aes-gcm コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --iv
 0xf90613bb8e337ec0339aad21 --data xvslgrtg8kHzrvekny97tLSIeokpPwV8
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e4",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
```

```
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --iv
 0xf90613bb8e337ec0339aad21 --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e4",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
```

```
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のスペース区切りリスト。

必須: いいえ

<AAD>

AES GCM 追加認証データ (AAD) 値 (16 進数)。

必須: いいえ

<TAG_LENGTH_BITS>

AES GCM タグの長さ (ビット単位)。

必須: はい

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: aes、des3、ec-private、generic-secret、rsa-private]。

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<IV>

キーを 16 進数でラッピングするために使用される初期値。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して AES-NO-PAD でキーのラッピングを解除する

CloudHSM CLI の key unwrap aes-no-pad コマンドを使用して、AES ラップキーとラップ解除メカニズムを使用してペイロードキーを AWS CloudHSM クラスターにラップAES-NO-PAD解除します。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、`local` 属性は `false` に設定されます。

key unwrap aes-no-pad コマンドを使用するには、 AWS CloudHSM クラスターに AES ラッピングキーがあり、そのunwrap属性を に設定する必要がありますtrue。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key unwrap aes-no-pad
Usage: key unwrap aes-no-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
```

```
        Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error
    --filter [<FILTER>...]
        Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
to unwrap with
    --data-path <DATA_PATH>
        Path to the binary file containing the wrapped key data
    --data <DATA>
        Base64 encoded wrapped key data
    --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
        Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
    --share-crypto-users [<SHARE_CRYPTO_USERS;...]
        Space separated list of Crypto User usernames to share the unwrapped key with
    --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
        The quorum value for key management operations for the unwrapped key
    --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
        The quorum value for key usage operations for the unwrapped key
    --key-type-class <KEY_TYPE_CLASS>
        Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
    --label <LABEL>
        Label for the unwrapped key
    --session
        Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
    --approval <APPROVAL>
        Filepath of signed quorum token file to approve operation
 -h, --help
        Print help
```

例

これらの例は、unwrap 属性値を true に設定した AES キーを使用して key unwrap aes-no-pad コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data eXK3PMAOnKM9y3YX6brbhtMoC060EOH9
```

```
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
```

```
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
```

```
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のスペース区切りリスト。

必須: いいえ

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: aes、des3、ec-private、generic-secret、rsa-private]。

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して AES-PKCS5-PAD でキーのラッピングを解除する

CloudHSM CLI の key unwrap aes-pkcs5-pad コマンドを使用して、AES ラッピングキーと AES-PKCS5-PAD ラップ解除メカニズムを使用してペイロードキーのラップを解除します。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、local 属性は false に設定されます。

key unwrap aes-pkcs5-pad コマンドを使用するには、 AWS CloudHSM クラスターに AES ラッピングキーがあり、そのunwrap属性を に設定する必要がありますtrue。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key unwrap aes-pkcs5-pad
Usage: key unwrap aes-pkcs5-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --share-crypto-users [<SHARE_CRYPTO_USERS;...]
          Space separated list of Crypto User usernames to share the unwrapped key with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
          The quorum value for key management operations for the unwrapped key
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
          The quorum value for key usage operations for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
```

```
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
  -h, --help
          Print help
```

例

これらの例は、unwrap 属性値を true に設定した AES キーを使用して key unwrap aes-pkcs5-pad コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY=
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
```

```
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
```

```
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。

必須: いいえ

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: `aes`、`des3`、`ec-private`、`generic-secret`、`rsa-private`]。

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して AES-ZERO-PAD でキーのラッピングを解除する

CloudHSM CLI の key unwrap aes-zero-pad コマンドを使用して、AES ラップキーとラップ解除メカニズムを使用してペイロードキーを AWS CloudHSM クラスターにラップAES-ZERO-PAD解除します。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、`local` 属性は `false` に設定されます。

key unwrap aes-zero-pad コマンドを使用するには、 AWS CloudHSM クラスターに AES ラッピングキーがあり、そのunwrap属性を に設定する必要がありますtrue。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key unwrap aes-zero-pad
Usage: key unwrap aes-zero-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
```

```
        Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error
    --filter [<FILTER>...]
        Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
to unwrap with
    --data-path <DATA_PATH>
        Path to the binary file containing the wrapped key data
    --data <DATA>
        Base64 encoded wrapped key data
    --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
        Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
    --share-crypto-users [<SHARE_CRYPTO_USERS;...]
        Space separated list of Crypto User usernames to share the unwrapped key with
    --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
        The quorum value for key management operations for the unwrapped key
    --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
        The quorum value for key usage operations for the unwrapped key
    --key-type-class <KEY_TYPE_CLASS>
        Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
    --label <LABEL>
        Label for the unwrapped key
    --session
        Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
    --approval <APPROVAL>
        Filepath of signed quorum token file to approve operation
 -h, --help
        Print help
```

例

これらの例は、unwrap 属性値を true に設定した AES キーを使用して key unwrap aes-zero-pad コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt
```

```
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
```

```
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
```

```
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のスペース区切りリスト。

必須: いいえ

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: aes、des3、ec-private、generic-secret、rsa-private]。

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して CLOUDHSM-AES-GCM でキーのラッピングを解除する

CloudHSM CLI の key unwrap cloudhsm-aes-gcm コマンドを使用して、AES ラップキーとラップ解除メカニズムを使用してペイロードキーを AWS CloudHSM クラスターにラップCLOUDHSM-AES-GCM解除します。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、`local` 属性は `false` に設定されます。

key unwrap cloudhsm-aes-gcm コマンドを使用するには、 AWS CloudHSM クラスターに AES ラッピングキーがあり、そのunwrap属性を に設定する必要がありますtrue。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key unwrap cloudhsm-aes-gcm
Usage: key unwrap cloudhsm-aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-
bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> <--data-
path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --share-crypto-users [<SHARE_CRYPTO_USERS;...]
          Space separated list of Crypto User usernames to share the unwrapped key with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
```

```
          The quorum value for key management operations for the unwrapped key
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
          The quorum value for key usage operations for the unwrapped key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
  -h, --help
          Print help
```

例

これらの例は、unwrap 属性値を true に設定した AES キーを使用して key unwrap cloudhsm-aes-gcm コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-
unwrapped --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --data
 6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
```

```
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --data-path payload-
key.pem
{
```

```
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
```

```
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。

必須: いいえ

<AAD>

AES GCM 追加認証データ (AAD) 値 (16 進数)。

必須: いいえ

<TAG_LENGTH_BITS>

AES GCM タグの長さ (ビット単位)。

必須: はい

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: aes、des3、ec-private、generic-secret、rsa-private]。

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して RSA-AES でキーをアンラップする

CloudHSM CLI の key unwrap rsa-aes コマンドを使用して、RSA プライベートキーと RSA-AES ラップ解除メカニズムを使用してペイロードキーをラップ解除します。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、local 属性は false に設定されます。

を使用するにはkey unwrap rsa-aes、 AWS CloudHSM クラスターに RSA パブリックラッピングキーの RSA プライベートキーがあり、そのunwrap属性を に設定する必要がありますtrue。

> **Note**
>
> このコマンドは、CloudHSM CLI 5.11 以降でのみ使用できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key unwrap rsa-aes
Usage: key unwrap rsa-aes [OPTIONS] --filter [<FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>
 <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --share-crypto-users [<SHARE_CRYPTO_USERS;...]
          Space separated list of Crypto User usernames to share the unwrapped key with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
```

```
          The quorum value for key management operations for the unwrapped key
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
          The quorum value for key usage operations for the unwrapped key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
  -h, --help
          Print help
```

例

これらの例は、unwrap 属性値が true に設定された RSA プライベートキーを使用して key unwrap rsa-aes コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped
 --filter attr.label=rsa-private-key-example --hash-function sha256 --
mgf mgf1-sha256 --data HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e2",
      "key-info": {
        "key-owners": [
          {
```

```
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-key-example --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
```

```
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。

必須: いいえ

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: `aes`、`des3`、`ec-private`、`generic-secret`、`rsa-private`]。

必須: はい

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1
- sha224
- sha256
- sha384
- sha512

必須: はい

<MGF>

マスク生成関数を指定します。

> **Note**
> マスク生成関数のハッシュ関数は、署名メカニズムのハッシュ関数と一致する必要があります。

有効な値:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI で RSA-OAEP でキーをアンラップする

CloudHSM CLI の key unwrap rsa-oaep コマンドを使用して、RSA プライベートキーと RSA-OAEP ラップ解除メカニズムを使用してペイロードキーをラップ解除します。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、`local` 属性は `false` に設定されます。

key unwrap rsa-oaep コマンドを使用するには、 AWS CloudHSM クラスターに RSA パブリックラッピングキーの RSA プライベートキーがあり、そのunwrap属性を に設定する必要がありますtrue。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key unwrap rsa-oaep
Usage: key unwrap rsa-oaep [OPTIONS] --filter [<FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>
 <--data-path <DATA_PATH>|--data <DATA>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --share-crypto-users [<SHARE_CRYPTO_USERS;...]
          Space separated list of Crypto User usernames to share the unwrapped key with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
          The quorum value for key management operations for the unwrapped key
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
          The quorum value for key usage operations for the unwrapped key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
```

```
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
  -h, --help
          Print help
```

例

これらの例は、unwrap 属性値が true に設定された RSA プライベートキーを使用して key unwrap rsa-oaep コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data
 OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/jGkDNdB4qyTAOQwEpggGf6v
+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d+F2KOONXsSxMhmxzzNG/
gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw==
{
  "data": {
    "key": {
      "key-reference": "0x00000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
```

```
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
```

```
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。

必須: いいえ

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: `aes`、`des3`、`ec-private`、`generic-secret`、`rsa-private`]。

必須: はい

<HASH_FUNCTION>

ハッシュ関数を指定します。

有効な値:

- sha1
- sha224
- sha256
- sha384
- sha512

必須: はい

<MGF>

マスク生成関数を指定します。

> **Note**
>
> マスク生成関数のハッシュ関数は、署名メカニズムのハッシュ関数と一致する必要があります。

有効な値:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド

- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して RSA-PKCS でキーをアンラップする

CloudHSM CLI の key unwrap rsa-pkcs コマンドを使用して、RSA プライベートキーと RSA-PKCS ラップ解除メカニズムを使用してペイロードキーをラップ解除します。

ラップされていないキーは、 によって生成されたキーと同じ方法で使用できます AWS CloudHSM。ローカルで生成されなかったことを示すために、local 属性は false に設定されます。

key unwrap rsa-pkcs コマンドを使用するには、 AWS CloudHSM クラスターに RSA パブリックラッピングキーの RSA プライベートキーがあり、そのunwrap属性を に設定する必要がありますtrue。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

構文

```
aws-cloudhsm > help key unwrap rsa-pkcs
Usage: key unwrap rsa-pkcs [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
```

```
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --share-crypto-users [<SHARE_CRYPTO_USERS;...]
          Space separated list of Crypto User usernames to share the unwrapped key with
      --manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
          The quorum value for key management operations for the unwrapped key
      --use-key-quorum-value <USE_KEY_QUORUM_VALUE;
          The quorum value for key usage operations for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
  -h, --help
          Print help
```

例

これらの例は、unwrap 属性値を true に設定した AES キーを使用して key unwrap rsa-oaep コマンドを使用する方法を示しています。

Example例: Base64 でエンコードされたラッピングされたキーデータからペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label
 aes-unwrapped --filter attr.label=rsa-private-key-example --data
 am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ef",
      "key-info": {
```

```
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example例: データパスを介して提供されたペイロードキーのラッピングを解除する

```
aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-key-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
```

```
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<FILTER>

キーリファレンス (例: `key-reference=0xabc`) または `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリストを使用して、ラッピングを解除するキーを選択します。

必須: はい

<DATA_PATH>

ラッピングされたキーデータを含むバイナリファイルへのパス。

必須: はい (Base64 でエンコードされたデータを通じて提供される場合を除く)

<DATA>

Base64 でエンコードされたラッピングされたキーデータ。

必須: はい (データパスを通じて提供される場合を除く)

<ATTRIBUTES>

ラッピングされたキーの `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のスペース区切りリスト。

必須: いいえ

<KEY_TYPE_CLASS>

ラッピングされたキーのキータイプとクラス [可能な値: aes、des3、ec-private、generic-secret、rsa-private]。

必須: はい

<LABEL>

ラッピングされていないキーのラベル。

必須: はい

<SESSION>

現在のセッションにのみ存在するセッションキーを作成します。セッション終了後、キーをリカバリすることはできません。

必須: いいえ

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラップ解除キーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI のキーラップコマンド

CloudHSM CLI の key wrap コマンドでは、対称または非対称プライベートキーの暗号化されたコピーをハードウェアセキュリティモジュール (HSM) からファイルにエクスポートします。key wrap を実行するときは、エクスポートするキーと出力ファイルの 2 つを指定します。エクスポートするキーは、エクスポートするキーを暗号化 (ラップ) する HSM 上のキーです。

key wrap コマンドは、HSM からキーを削除したり、暗号化操作での使用を妨げたりしません。同じキーを複数回エクスポートできます。暗号化されたキーを HSM に再度インポートするには、CloudHSM CLI のキーアンラップコマンド を使用します。キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、暗号化オペレーションでのみキーを使用できます

key wrap コマンドは次のサブコマンドで構成されます。

- aes-gcm
- aes-no-pad
- aes-pkcs5-pad
- aes-zero-pad
- cloudhsm-aes-gcm
- rsa-aes
- rsa-oaep
- rsa-pkcs

CloudHSM CLI を使用して AES-GCM でキーをラップする

CloudHSM CLI の key wrap aes-gcm コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の AES キーと `AES-GCM` ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの `extractable` 属性を `true` に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap aes-gcm コマンドを使用するには、まず AWS CloudHSM クラスターに AES キーが必要です。CloudHSM CLI で対称 AES キーを生成する コマンドと `true` に設定された `wrap` 属性でラップするための AES キーを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap aes-gcm
```

```
Usage: key wrap aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
  -h, --help
          Print help
```

例

この例では、AES キーを使用して key wrap aes-gcm コマンドを使用する方法を示しています。

Example

```
aws-cloudhsm > key wrap aes-gcm --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example --tag-length-bits 64  --aad 0x10
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "iv": "0xf90613bb8e337ec0339aad21",
    "wrapped_key_data": "xvslgrtg8kHzrvekny97tLSIeokpPwV8"
```

```
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<AAD>

AES GCM 追加認証データ (AAD) 値 (16 進数)。

必須: いいえ

<TAG_LENGTH_BITS>

AES GCM タグの長さ (ビット単位)。

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI でキーを AES-NO-PAD でラップする

CloudHSM CLI の key wrap aes-no-pad コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の AES キーと AES-NO-PAD ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの extractable 属性を true に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap aes-no-pad コマンドを使用するには、まず AWS CloudHSM クラスターに AES キーが必要です。CloudHSM CLI で対称 AES キーを生成する コマンドと true に設定された wrap 属性を使用して、ラッピング用の AES キーを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap aes-no-pad
Usage: key wrap aes-no-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...]
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
  -h, --help
          Print help
```

例

この例では、wrap 属性値が true に設定された AES キーを使用して key wrap aes-no-pad コマンドを使用する方法を示します。

Example

```
aws-cloudhsm > key wrap aes-no-pad --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "eXK3PMAOnKM9y3YX6brbhtMoC060EOH9"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド

- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して AES-PKCS5-PAD でキーをラップする

CloudHSM CLI の key wrap aes-pkcs5-pad コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の AES キーと AES-PKCS5-PAD ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの extractable 属性を true に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap aes-pkcs5-pad コマンドを使用するには、まず AWS CloudHSM クラスターに AES キーが必要です。CloudHSM CLI で対称 AES キーを生成する コマンドと true に設定された wrap 属性を使用して、ラッピング用の AES キーを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap aes-pkcs5-pad
Usage: key wrap aes-pkcs5-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
```

```
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
  -h, --help
          Print help
```

例

この例では、wrap 属性値が true に設定された AES キーを使用して key wrap aes-pkcs5-pad コマンドを使用する方法を示します。

Example

```
aws-cloudhsm > key wrap aes-pkcs5-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY="
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のキーリファレンス (例: `key-reference=0xabc`) またはスペース区切りリスト。

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI でキーを AES-ZERO-PAD でラップする

CloudHSM CLI の key wrap aes-zero-pad コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の AES キーと `AES-ZERO-PAD` ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの `extractable` 属性を `true` に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap aes-zero-pad コマンドを使用するには、まず AWS CloudHSM クラスターに AES キーが必要です。CloudHSM CLI で対称 AES キーを生成する コマンドで `wrap` 属性を `true` に設定して、ラッピング用の AES キーを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap aes-zero-pad
Usage: key wrap aes-zero-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
  -h, --help
          Print help
```

例

この例では、wrap 属性値が true に設定された AES キーを使用して key wrap aes-zero-pad コマンドを使用する方法を示します。

Example

```
aws-cloudhsm > key wrap aes-zero-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して CLOUDHSM-AES-GCM でキーをラップする

CloudHSM CLI の key wrap cloudhsm-aes-gcm コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の AES キーと `CLOUDHSM-AES-GCM` ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの `extractable` 属性を `true` に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap cloudhsm-aes-gcm コマンドを使用するには、まず AWS CloudHSM クラスターに AES キーが必要です。CloudHSM CLI で対称 AES キーを生成する コマンドと `true` に設定された `wrap` 属性でラップするための AES キーを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap cloudhsm-aes-gcm
Usage: key wrap cloudhsm-aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
  -h, --help
          Print help
```

例

この例では、AES キーを使用して key wrap cloudhsm-aes-gcm コマンドを使用する方法を示しています。

Example

```
aws-cloudhsm > key wrap cloudhsm-aes-gcm --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example --tag-length-bits 64 --aad 0x10
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ"
  }
```

```
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のキーリファレンス (例: `key-reference=0xabc`) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式のキー属性のキーリファレンス (例: `key-reference=0xabc`) またはスペース区切りリスト。

必須: はい

<AAD>

AES GCM 追加認証データ (AAD) 値 (16 進数)。

必須: いいえ

<TAG_LENGTH_BITS>

AES GCM タグの長さ (ビット単位)。

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI で RSA-AES を使用してキーをラップする

CloudHSM CLI の key wrap rsa-aes コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の RSA パブリックキーと RSA-AES ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの `extractable` 属性を `true` に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap rsa-aes コマンドを使用するには、まず AWS CloudHSM クラスターに RSA キーが必要です。CloudHSM CLI の generate-asymmetric-pair カテゴリ コマンドと `true` に設定された `wrap` 属性を使用して、RSA キーペアを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap rsa-aes
Usage: key wrap rsa-aes [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>
```

```
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
  -h, --help
          Print help
```

例

この例では、wrap 属性値を true に設定した RSA 公開キーを使用して key wrap rsa-ae コマンドを使用する方法を示します。

Example

```
aws-cloudhsm > key wrap rsa-aes --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
```

```
    "wrapped-key-data": "HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg=="
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<MGF>

マスク生成関数を指定します。

> **Note**
>
> マスク生成関数のハッシュ関数は、署名メカニズムのハッシュ関数と一致する必要があります。

有効値

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して RSA-OAEP でキーをラップする

CloudHSM CLI の key wrap rsa-oaep コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の RSA パブリックキーと RSA-OAEP ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの `extractable` 属性を `true` に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap rsa-oaep コマンドを使用するには、まず AWS CloudHSM クラスターに RSA キーが必要です。CloudHSM CLI の generate-asymmetric-pair カテゴリ コマンドと `true` に設定された `wrap` 属性を使用して、RSA キーペアを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap rsa-oaep
Usage: key wrap rsa-oaep [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
mgf1-sha256, mgf1-sha384, mgf1-sha512]
  -h, --help
          Print help
```

例

この例では、wrap 属性値を true に設定した RSA 公開キーを使用して key wrap rsa-oaep コマンドを使用する方法を示します。

Example

```
aws-cloudhsm > key wrap rsa-oaep --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/
jGkDNdB4qyTAOQwEpggGf6v+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d
+F2KOONXsSxMhmxzzNG/gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/
wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw=="
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<MGF>

マスク生成関数を指定します。

> **Note**
>
> マスク生成関数のハッシュ関数は、署名メカニズムのハッシュ関数と一致する必要があります。

有効値

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド

- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI で RSA-PKCS でキーをラップする

CloudHSM CLI の key wrap rsa-pkcs コマンドを使用して、ハードウェアセキュリティモジュール (HSM) の RSA パブリックキーと RSA-PKCS ラップメカニズムを使用してペイロードキーをラップします。ペイロードキーの extractable 属性を true に設定する必要があります。

キーの所有者、つまりキーを作成した Crypto User (CU) のみがキーをラップできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用できます。

key wrap rsa-pkcs コマンドを使用するには、まず AWS CloudHSM クラスターに RSA キーが必要です。CloudHSM CLI の generate-asymmetric-pair カテゴリ コマンドと true に設定された wrap 属性を使用して、RSA キーペアを生成できます。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

要件

- このコマンドを実行するには、CU としてログインする必要があります。

Syntax

```
aws-cloudhsm > help key wrap rsa-pkcs
Usage: key wrap rsa-pkcs [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
```

```
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --wrapping-approval <WRAPPING_APPROVALR>
          File path of signed quorum token file to approve operation for wrapping key
      --payload-approval <PAYLOAD_APPROVALR>
          File path of signed quorum token file to approve operation for payload key
  -h, --help
          Print help
```

例

この例では、RSA パブリックキーを使用して key wrap rsa-pkcs コマンドを使用する方法を示しています。

Example

```
aws-cloudhsm > key wrap rsa-pkcs --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000007008da",
    "wrapped_key_data": "am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp
+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg=="
  }
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PAYLOAD_FILTER>

ペイロードキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<PATH>

ラップされたキーデータを保存するバイナリファイルへのパス。

必須: いいえ

<WRAPPING_FILTER>

ラッピングキーを選択する attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 形式のキー属性のキーリファレンス (例: key-reference=0xabc) またはスペース区切りリスト。

必須: はい

<WRAPPING_APPROVALR>

ラッピングキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ラッピングキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

<PAYLOAD_APPROVALR>

ペイロードキーのオペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。ペイロードキーのキー管理サービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- CloudHSM CLI のキーラップコマンド
- CloudHSM CLI のキーアンラップコマンド

CloudHSM CLI を使用して HSM にログインする

CloudHSM CLI の login コマンドを使用して、 AWS CloudHSM クラスター内の各ハードウェアセキュリティ (HSM) にログインおよびログアウトできます。このコマンドは、次のサブコマンドで構成されます。

- mfa-token-sign

> **Note**
>
> ログイン試行回数が 5 回を超えると、アカウントがロックアウトされます。アカウントのロックを解除するには、管理者は、cloudhsm_cli の user change-password コマンドを使用してパスワードをリセットする必要があります。

ログインとログアウトのトラブルシューティングを行うには

クラスター内に複数の HSM がある場合は、アカウントがロックアウトされるまでのログイン試行回数の上限が増える可能性があります。これは、CloudHSM クライアントがさまざまな HSM 間で負荷を分散するためです。したがって、ログイン試行は毎回同じ HSM で開始されない場合があります。この機能をテストしている場合は、アクティブな HSM が1つだけのクラスターでテストすることをお勧めします。

2018 年 2 月より前にクラスターを作成した場合、ロックアウトされるまでのログイン試行回数は 20 回です。

ユーザーのタイプ

これらのコマンドは、次のユーザーが実行できます。

- 非アクティブ管理者
- 管理者
- Crypto User (CU)

Syntax

```
aws-cloudhsm > help login
Login to your cluster

USAGE:
    cloudhsm-cli login [OPTIONS] --username <USERNAME> --role <ROLE> [COMMAND]

Commands:
  mfa-token-sign  Login with token-sign mfa
  help            Print this message or the help of the given subcommand(s)
```

```
OPTIONS:
        --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

        --username <USERNAME>
            Username to access the Cluster

        --role <ROLE>
            Role the user has in the Cluster

           Possible values:
           - crypto-user: A CryptoUser has the ability to manage and use keys
           - admin:       An Admin has the ability to manage user accounts

        --password <PASSWORD>
           Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

  -h, --help
          Print help (see a summary with '-h')
```

例

Example

このコマンドは、admin1 という名前の管理者ユーザーの認証情報を使用して、クラスター内のすべての HSM にログインします。

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<USERNAME>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。このコマンドではユーザー名の大文字と小文字は区別されません。ユーザー名は常に小文字で表示されます。

必須: はい

<ROLE>

このユーザーに割り当てられるロールを指定します。有効な値は admin、crypto-user です。

ユーザーのロールを取得するには、 user list コマンドを使用します。HSM のユーザータイプの詳細については、「HSM ユーザーについて」を参照してください。

必須: はい

<PASSWORD>

HSM にログインしているユーザーのパスワードを指定します。

必須: お客様は、`--password`コマンドライン引数を介してパスワードを指定するか、インタラクティブにプロンプトが表示されるように省略する必要があります。

> **Note**
>
> コマンドラインで `--password`引数を使用すると、シェル履歴に認証情報が公開される可能性があります。パスワードをインタラクティブにプロンプトするか、使用後にシェル履歴をクリアすることを検討してください。

関連トピック

- CloudHSM CLI の使用開始

- クラスターのアクティブ化

CloudHSM CLI を使用して MFA で HSM にログインする

AWS CloudHSM CloudHSM CLI の login mfa-token-sign コマンドを使用して、多要素認証 (MFA) を使用してハードウェアセキュリティモジュール (HSM) にログインします。このコマンドを使用するには、まず CloudHSM CLI の MFA を設定する必要があります。

ユーザーのタイプ

これらのコマンドは、次のユーザーが実行できます。

- 管理者
- Crypto User (CU)

Syntax

```
aws-cloudhsm > help login mfa-token-sign
Login with token-sign mfa

USAGE:
    login --username <username> --role <role> mfa-token-sign --token <token>

OPTIONS:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --token <TOKEN>            Filepath where the unsigned token file will be written
  -h, --help                     Print help
```

例

Example

```
aws-cloudhsm > login --username test_user --role admin mfa-token-sign --token /home/
valid.token
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
{
  "error_code": 0,
  "data": {
```

```
    "username": "test_user",
    "role": "admin"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<TOKEN>

署名されていないトークンファイルが書き込まれるファイルパス。

必須: はい

関連トピック

- CloudHSM CLI の使用開始
- クラスターのアクティブ化
- CloudHSM CLI を使用して MFA を管理する

CloudHSM CLI で HSM からのログアウトする

CloudHSM CLI の logout コマンドを使用して、 AWS CloudHSM クラスター内の各ハードウェアセキュリティモジュール (HSM) からログアウトします。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者
- Crypto User (CU)

Syntax

```
aws-cloudhsm > help logout
```

```
Logout of your cluster

USAGE:
    logout

OPTIONS:
        --cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
    -h, --help                    Print help information
    -V, --version                 Print version information
```

例

Example

このコマンドはクラスターの HSM からログアウトします。

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

関連トピック

- CloudHSM CLI の使用開始
- クラスターのアクティブ化

CloudHSM CLI のユーザーカテゴリ

CloudHSM CLI では、user はコマンドグループの親カテゴリであり、親カテゴリと組み合わせるとユーザー固有のコマンドが作成されます。現在、このユーザーカテゴリは次のコマンドで構成されています。

- user change-mfa
- user change-password
- user create
- user delete

- user list
- user replicate

CloudHSM CLI のユーザー change-mfa カテゴリ

CloudHSM CLI において、user change-mfa は、多要素認証 (MFA) の変更に特化したコマンドを作成するために、親カテゴリと組み合わせて使用されるコマンド群の親カテゴリです。

現在、このカテゴリは次のサブコマンドで構成されています。

- token-sign

CloudHSM CLI を使用してユーザーの MFA 設定を変更する

CloudHSM CLI で user change-mfa token-sign コマンドを使用して、ユーザーアカウントの多要素認証 (MFA) 設定を更新します。このコマンドは、どのユーザーアカウントでも実行できます。Admin ロールを持つアカウントは、他のユーザーに対してこのコマンドを実行できます。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者
- Crypto User

構文

現在、ユーザーが利用できる多要素戦略は「トークン署名」だけです。

```
aws-cloudhsm > help user change-mfa
Change a user's Mfa Strategy

Usage:
    user change-mfa <COMMAND>

Commands:
  token-sign  Register or Deregister a public key using token-sign mfa strategy
  help        Print this message or the help of the given subcommand(s)
```

トークン署名戦略では、署名されていないトークンを書き込むためのトークンファイルを要求します。

```
aws-cloudhsm > help user change-mfa token-sign
Register or Deregister a public key using token-sign mfa strategy

Usage: user change-mfa token-sign [OPTIONS] --username <USERNAME> --role <ROLE> <--
token <TOKEN>|--deregister>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username of the user that will be modified

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --change-password <CHANGE_PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --token <TOKEN>
          Filepath where the unsigned token file will be written. Required for enabling
 MFA for a user

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

      --deregister
          Deregister the MFA public key, if present

      --change-quorum
          Change the Quorum public key along with the MFA key

  -h, --help
```

```
          Print help (see a summary with '-h')
```

例

このコマンドは、クラスター内の HSM ごとに 1 つの未署名トークンを token で指定されたファイルに書き込みます。プロンプトが表示されたら、ファイル内のトークンに署名します。

Example: クラスターの HSM ごとに 1 つの署名なしトークンを書き込みます

```
aws-cloudhsm > user change-mfa token-sign --username cu1 --change-password password --
role crypto-user --token /path/myfile
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:/path/mypemfile
{
  "error_code": 0,
  "data": {
    "username": "test_user",
    "role": "admin"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<ROLE>

ユーザーアカウントに付与されるロールを指定します。このパラメータは必須です。HSM のユーザータイプの詳細については、「HSM ユーザーについて」を参照してください。

有効な値

- Admin:管理者はユーザーを管理できますが、キーを管理することはできません。
- Crypto user: Crypto User は、管理キーを作成し、暗号化オペレーションでキーを使用できます。

<USERNAME>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。

ユーザーの作成後にユーザー名を変更することはできません。CloudHSM CLI コマンドでは、ロールとパスワードでは大文字と小文字が区別されますが、ユーザー名では区別されません。

必須: はい

<CHANGE_PASSWORD>

MFA を登録/登録解除するユーザーのプレーンテキストの新しいパスワードを指定します。

必須: はい

<TOKEN>

署名なしトークンファイルが書き込まれるファイルパス。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。クォーラムユーザーサービスのクォーラム値が 1 より大きい場合にのみ必要です。

<DEREGISTER>

MFA パブリックキーが存在する場合、登録を解除します。

<CHANGE-QUORUM>

クォーラムパブリックキーを MFA キーと一緒に変更します。

関連トピック

- HSM ユーザー用 2FA について

CloudHSM CLI でユーザーのパスワードを変更する

CloudHSM CLI の user change-password コマンドを使用して、 AWS CloudHSM クラスター内の既存のユーザーのパスワードを変更します。ユーザーの MFA を有効にするには、`user change-mfa` コマンドを使用します。

どのユーザーも自分のパスワードを変更できます。さらに、管理者ロールを持つユーザーは、クラスター内の別のユーザーのパスワードを変更できます。変更するために現在のパスワードを入力する必要はありません。

> **Note**
>
> 現在クラスターにログインしているユーザーのパスワードは変更できません。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者
- Crypto User (CU)

構文

> **Note**
>
> ユーザーの多要素認証 (MFA) を有効にするには、user change-mfa コマンドを使用します。

```
aws-cloudhsm > help user change-password
Change a user's password

    Usage:
        cloudhsm-cli user change-password [OPTIONS] --username <USERNAME> --role <ROLE>
 [--password <PASSWORD>]

    Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username of the user that will be modified

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
```

```
          - admin:       An Admin has the ability to manage user accounts

      --password <PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

      --deregister-mfa <DEREGISTER-MFA>
          Deregister the user's mfa public key, if present

      --deregister-quorum <DEREGISTER-QUORUM>
          Deregister the user's quorum public key, if present
 -h, --help
          Print help (see a summary with '-h')
```

例

次の例は、user change-password を使用して、現在のユーザーまたはクラスター内の他のユーザーのパスワードをリセットする方法を示しています。

Example: パスワードの変更

クラスター内のすべてのユーザーは、user change-password を使用して自分のパスワードを変更できます。

次の出力は、Bob が現在 Crypto User (CU) としてログインしていることを示しています。

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。クォーラムユーザーサービスのクォーラム値が 1 より大きい場合にのみ必要です。

<DEREGISTER-MFA>

MFA パブリックキーが存在する場合、登録を解除します。

<DEREGISTER-QUORUM>

Quorum パブリックキーがある場合は、登録を解除します。

<PASSWORD>

ユーザーのプレーンテキストの新しいパスワードを指定します。「:」の文字は使用できません。

必須: はい

<ROLE>

ユーザーアカウントに付与されるロールを指定します。このパラメータは必須です。HSM のユーザータイプの詳細については、「HSM ユーザーについて」を参照してください。

有効な値

- Admin:管理者はユーザーを管理できますが、キーを管理することはできません。
- Crypto user: Crypto User は、管理キーを作成し、暗号化オペレーションでキーを使用できます。

<USERNAME>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。

ユーザーの作成後にユーザー名を変更することはできません。CloudHSM CLI コマンドでは、ロールとパスワードでは大文字と小文字が区別されますが、ユーザー名では区別されません。

必須: はい

関連トピック

- user list
- user create
- user delete

CloudHSM CLI のユーザークォーラム変更カテゴリ

CloudHSM CLI では、user change-quorum はコマンドグループの親カテゴリであり、これを親カテゴリと組み合わせると、ユーザーのクォーラム変更専用のコマンドが作成されます。

user change-quorum は指定されたクォーラム戦略を使用してユーザークォーラム認証を登録するために使用されます。SDK 5.8.0 では、次に示すように、ユーザーが利用できるクォーラム戦略は 1 つだけです。

現在、このカテゴリは次のカテゴリとサブコマンドで構成されています。

- token-sign
  - 登録

CloudHSM CLI の change-quorum token-sign カテゴリ

CloudHSM CLI では、user change-quorum token-sign はコマンドの親カテゴリで、この親カテゴリと組み合わせるとトークン署名クォーラムオペレーション専用のコマンドが作成されます。

現在、このカテゴリは以下のコマンドで構成されています。

- 登録

CloudHSM CLI を使用してユーザーのトークン署名クォーラム戦略を登録する

CloudHSM CLI の user change-quorum token-sign register コマンドを使用して、管理者ユーザーのトークン署名クォーラム戦略を登録します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者

Syntax

```
aws-cloudhsm > help user change-quorum token-sign register
Register a user for quorum authentication with a public key

Usage: user change-quorum token-sign register --public-key <PUBLIC_KEY> --signed-
token <SIGNED_TOKEN>

Options:
      --cluster-id <CLUSTER_ID>      Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --public-key <PUBLIC_KEY>      Filepath to public key PEM file
      --signed-token <SIGNED_TOKEN>  Filepath with token signed by user private key
  -h, --help Print help (see a summary with '-h')
```

例

Example

このコマンドを実行するには、register quorum token-sign を実行するユーザーとしてログインする必要があります。

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

user change-quorum token-sign register コマンドは HSM にパブリックキーを登録します。その結果、必要なクォーラム値のしきい値を満たすためにユーザーがクォーラム署名を取得する必要があるクォーラム必須オペレーションのクォーラム承認者としての資格が得られます。

```
aws-cloudhsm > user change-quorum token-sign register \
```

```
    --public-key /home/mypemfile \
    --signed-token /home/mysignedtoken
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

これで、user list コマンドを実行して、このユーザーにクォーラムトークン署名が登録されていることを確認できます。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin1",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<PUBLIC-KEY>

パブリックキー PEM ファイルへのファイルパス。

必須: はい

<SIGNED-TOKEN>

ユーザーのプライベートキーで署名されたトークンを含むファイルパス。

必須: はい

関連トピック

- CloudHSM CLI を使用してクォーラム認証を管理する
- 管理者用クォーラム認証を使用する: 初回セットアップ
- 管理者のクォーラム最小値を変更する
- クォーラム認証をサポートするサービス名とタイプ

CloudHSM CLI を使用して AWS CloudHSM ユーザーを作成する

CloudHSM CLI の user create コマンドは、 AWS CloudHSM クラスターにユーザーを作成します。このコマンドを実行できるのは、管理者ロールを持つユーザーアカウントのみです。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- 管理者

要件

このコマンドを実行するには、管理者ユーザーとしてログインする必要があります

Syntax

```
aws-cloudhsm > help user create
Create a new user

Usage: cloudhsm-cli user create [OPTIONS] --username <USERNAME> --role <ROLE> [--
password <PASSWORD>]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username to access the HSM cluster

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --password <PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

  -h, --help
          Print help (see a summary with '-h')
```

例

以下の例では、user create を使用して HSM に新しいユーザーを作成する方法を示します。

Example: Crypto User を作成する

この例では、 暗号化ユーザーロールを使用して AWS CloudHSM クラスターに アカウントを作成します。

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<USERNAME>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。このコマンドではユーザー名の大文字と小文字は区別されません。ユーザー名は常に小文字で表示されます。

必須: はい

<ROLE>

このユーザーに割り当てられるロールを指定します。このパラメータは必須です。有効な値は admin、crypto-user です。

ユーザーのロールを取得するには、user list コマンドを使用します。HSM のユーザー タイプの詳細については、「HSM ユーザーについて」を参照してください。

<PASSWORD>

HSM にログインしているユーザーのパスワードを指定します。「:」の文字は使用できません。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。クォーラムユーザーサービスのクォーラム値が 1 より大きい場合にのみ必要です。

関連トピック

- user list
- user delete
- user change-password

CloudHSM CLI で AWS CloudHSM ユーザーを削除する

CloudHSM CLI の user delete コマンドは、 AWS CloudHSM クラスターからユーザーを削除します。このコマンドを実行できるのは、管理者ロールを持つユーザーアカウントのみです。現在 HSM にログインしているユーザーを削除することはできません。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- 管理者

要件

- キーを所有しているユーザーアカウントを削除することはできません。
- このコマンドを実行するには、ユーザーアカウントに管理者ロールが必要です。

構文

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
aws-cloudhsm > help user delete
Delete a user

Usage: user delete [OPTIONS] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
```

```
          Username to access the HSM cluster

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
```

例

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<USERNAME>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。このコマンドではユーザー名の大文字と小文字は区別されません。ユーザー名は常に小文字で表示されます。

必須: はい

<ROLE>

このユーザーに割り当てられるロールを指定します。このパラメータは必須です。有効な値は admin、crypto-user です。

ユーザーのロールを取得するには、user list コマンドを使用します。HSM のユーザータイプの詳細については、「HSM ユーザーについて」を参照してください。

必須: はい

<APPROVAL>

オペレーションを承認する署名付きクォーラムトークンファイルへのファイルパスを指定します。クォーラムユーザーサービスのクォーラム値が 1 より大きい場合にのみ必要です。

必須: はい

関連トピック

- user list
- user create
- user change-password

CloudHSM CLI を使用するすべての AWS CloudHSM ユーザーを一覧表示する

CloudHSM CLI の user list コマンドは、 AWS CloudHSM クラスターに存在するユーザーアカウントを一覧表示します。このコマンドは、CloudHSM CLI にログインしていなくても実行できます。

> **Note**
>
> HSMs を追加または削除する場合は、 AWS CloudHSM クライアントとコマンドラインツールが使用する設定ファイルを更新します。そうしないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- すべてのユーザー。このコマンドは、ログインしていなくても実行できます。

Syntax

```
aws-cloudhsm > help user list
List the users in your cluster
```

```
USAGE:
    user list

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

例

このコマンドは、CloudHSM クラスターに存在するユーザーを一覧表示します。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "test_user",
        "role": "admin",
        "locked": "false",
        "mfa": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
```

```
      }
    ]
  }
}
```

この出力が示すユーザー属性は以下のとおりです。

- Username: ユーザー定義のわかりやすいユーザー名を表示します。ユーザー名は常に小文字で表示されます。
- Role: HSM でユーザーが実行できるオペレーションを決定します。
- Locked: このユーザーアカウントがロックアウトされているかどうかを示します。
- MFA: このユーザーアカウントでサポートされている多要素認証メカニズムを示します。
- Cluster coverage: このユーザーアカウントのクラスター全体での可用性を示します。

関連トピック

- key_mgmt_util で listUsers
- user create
- user delete
- user change-password

CloudHSM CLI でユーザーをレプリケートする

CloudHSM CLI の user replicate コマンドを使用して、ユーザーをソース AWS CloudHSM クラスターから宛先 AWS CloudHSM クラスターにレプリケートします。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- 管理者 (CO)

要件

- 送信元クラスターと送信先クラスターはクローンである必要があります。つまり、一方が他方のバックアップから作成されたか、どちらも共通のバックアップから作成されたということです。詳細については「バックアップからクラスターを作成する」を参照してください。

- このコマンドを実行するには、送信元クラスターと送信先クラスターの両方で、管理者としてログインしている必要があります。
  - 単一コマンドモードでは、コマンドは CLOUDHSM_PIN および CLOUDHSM_ROLE 環境変数を使用してソースクラスターで認証します。詳細については「シングルコマンドモード」を参照してください。送信先クラスターの認証情報を提供するには、DESTINATION_CLOUDHSM_PIN と DESTINATION_CLOUDHSM_ROLE の 2 つの追加の環境変数を設定する必要があります。

```
$ export DESTINATION_CLOUDHSM_ROLE=<role>
```

```
$ export DESTINATION_CLOUDHSM_PIN=<username:password>
```

  - インタラクティブモードでは、ユーザーは送信元クラスターと送信先クラスターの両方に明示的にログインする必要があります。

構文

```
aws-cloudhsm > help user replicate
Replicate a user from a source to a destination cluster

Usage: user replicate --username <USERNAME> --role <ROLE> --source-cluster-
id <SOURCE_CLUSTER_ID> --destination-cluster-id <DESTINATION_CLUSTER_ID>

Options:
      --username <USERNAME>
          Username of the user to replicate

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --source-cluster-id <SOURCE_CLUSTER_ID>
          Source cluster ID

      --destination-cluster-id <DESTINATION_CLUSTER_ID>
          Destination cluster ID

  -h, --help
```

```
          Print help (see a summary with '-h')
```

例

Example例: ユーザーをレプリケートする

このコマンドは、送信元のクラスターから送信先クラスター (クローン) にユーザーをレプリケートします。以下の例は、両方のクラスターに管理者としてログインしたときの出力を示しています。

```
admin-user@cluster-1234abcdefg > user replicate \
      --username example-admin \
      --role admin \
      --source-cluster-id cluster-1234abcdefg \
      --destination-cluster-id cluster-2345bcdefgh
{
  "error_code": 0,
  "data": {
    "user": {
      "username": "example-admin",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [],
      "cluster-coverage": "full"
    },
    "message": "Successfully replicated user"
  }
}
```

引数

<USERNAME>

ソースクラスターでレプリケートするユーザーのユーザー名を指定します。

必須: はい

<ROLE>

このユーザーに割り当てられるロールを指定します。このパラメータは必須です。有効な値は admin、crypto-user です。

ユーザーのロールを取得するには、user list コマンドを使用します。HSM のユーザータイプの詳細については、「HSM ユーザーについて」を参照してください。

必須: はい

<SOURCE_CLUSTER_ID>

送信元クラスターの ID。

必須: はい

<DESTINATION_CLUSTER_ID>

送信先クラスターの ID。

必須: はい

関連トピック

- CloudHSM CLI を使用した複数のクラスターへの接続

CloudHSM CLI のクォーラムカテゴリ

CloudHSM CLIにおいて、quorum は、コマンドのグループの親カテゴリであり、これらと quorum を組み合わせることで、クォラム認証や、M of N オペレーションに固有のコマンドを作成できます。現在、このカテゴリは独自のコマンドで構成される token-sign サブカテゴリで構成されています。詳細については、以下のリンクをクリックしてください。

- token-sign

管理サービス: クォーラム認証は、ユーザーの作成、ユーザーの削除、ユーザーパスワードの変更、クォーラム値の設定、クォーラム機能と MFA 機能の無効化などの管理者権限を持つサービスに使用されます。

Crypto User サービス: クォーラム認証は、特定のキーに関連する Crypt User の特権サービス (キーによる署名、キーの共有/共有解除)、キーのラッピング/ラップ解除、キー属性の設定など) に使用されます。関連付けられたキーのクォーラム値は、そのキーを生成、インポート、またはラップ解除する際に設定されます。クォーラム値は、そのキーが関連付けられているユーザー数 (キーが共有されているユーザーおよびキー所有者を含む) 以下である必要があります。

各サービスタイプはさらに適格なサービス名に分類されます。このサービス名には、実行可能なクォーラムがサポートする特定のサービスオペレーションのセットが含まれます。

サービス名	サービスタイプ	サービスオペレーション
ユーザー	管理者	• user create • user delete • user change-password • user change-mfa
quorum	管理者	• quorum token-sign set-quorum-value
cluster[1]	管理者	• cluster mtls register-trust-anchor • cluster mtls deregister-trust-anchor • cluster mtls set-enforcement
キーの管理	Crypto User	• キーのラップ • キーのラップ解除 • キーシェア • キー共有解除 • key set-attribute
キーの用途	Crypto User	• キーサイン

[1] クラスターサービスは hsm2m.medium でのみ利用できます

関連トピック

- CloudHSM CLI を使用して AWS CloudHSM 管理者のクォーラム認証を設定する
- CloudHSM CLI を使用したクォーラム認証の管理 (M of N アクセスコントロール)

CloudHSM CLI の quorum token-sign カテゴリ

CloudHSM CLI において、quorum token-sign はコマンド グループのカテゴリであり、quorum token-sign と組み合わせると、クォーラム認証や M of N オペレーションに固有のコマンドを作成できます。

現在、このカテゴリは次のコマンドで構成されています。

- 削除
- 生成
- リスト
- list-quorum-values
- set-quorum-value

CloudHSM CLI を使用してクォーラムトークンを削除する

CloudHSM CLI の quorum token-sign delete のコマンドを使用して、クォーラム承認サービスの 1 つ以上のトークンを削除します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者

Syntax

```
aws-cloudhsm > help quorum token-sign delete
Delete one or more Quorum Tokens

Usage: quorum token-sign delete --scope <SCOPE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --scope <SCOPE>
          Scope of which token(s) will be deleted

          Possible values:
          - user: Deletes all token(s) of currently logged in user
          - all:  Deletes all token(s) on the HSM
  -h, --help
          Print help (see a summary with '-h')
```

例

次の例は、CloudHSM CLI の quorum token-sign delete コマンドを使用して、クォーラム認定サービスの 1 つ以上のトークンを削除する方法を示しています。

Example: クォーラム認定サービスの 1 つ以上のトークンを削除します

```
aws-cloudhsm > quorum token-sign delete --scope all
{
  "error_code": 0,
  "data": "Deletion of quorum token(s) successful"
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<SCOPE>

AWS CloudHSM クラスター内でトークン (複数可) が削除されるスコープ。

有効な値

- ユーザー: ログインしているユーザーが所有するトークンのみを削除する場合に使用します。
- すべて: AWS CloudHSM クラスター内のすべてのトークンを削除するために使用します。

関連トピック

- user list
- user create
- user delete

CloudHSM CLI を使用してクォーラムトークンを生成する

CloudHSM CLI の quorum token-sign generate コマンドを使用して、クォーラム承認サービスのトークンを生成します。

サービスユーザーとクォーラムの HSM クラスターでは、サービスごとに 1 ユーザーにつき 1 つのアクティブトークンを取得することには制限があります。この制限は、キーサービスに関連するトークンには適用されません。

> **Note**
>
> 管理者と Crypto User のみが、特定のサービストークンを生成できます。サービスのタイプと名前について詳しくは、クォーラム認証をサポートするサービス名とタイプを参照してください。

管理サービス: クォーラム認証は、ユーザーの作成、ユーザーの削除、ユーザーパスワードの変更、クォーラム値の設定、クォーラム機能と MFA 機能の無効化などの管理者権限を持つサービスに使用されます。

Crypto User サービス: クォーラム認証は、特定のキーに関連する Crypt User の特権サービス (キーによる署名、キーの共有/共有解除)、キーのラッピング/ラップ解除、キー属性の設定など) に使用されます。関連付けられたキーのクォーラム値は、そのキーを生成、インポート、またはラップ解除する際に設定されます。クォーラム値は、そのキーが関連付けられているユーザー数 (キーが共有されているユーザーおよびキー所有者を含む) 以下である必要があります。

各サービスタイプはさらに適格なサービス名に分類されます。このサービス名には、実行可能なクォーラムがサポートする特定のサービスオペレーションのセットが含まれます。

サービス名	サービスタイプ	サービスオペレーション
ユーザー	管理者	• user create • user delete • user change-password • user change-mfa
quorum	管理者	• quorum token-sign set-quorum-value
cluster[1]	管理者	• cluster mtls register-trust-anchor • cluster mtls deregister-trust-anchor

サービス名	サービスタイプ	サービスオペレーション
		• cluster mtls set-enforcement
キーの管理	Crypto User	• キーのラップ • キーのラップ解除 • キーシェア • キー共有解除 • key set-attribute
キーの用途	Crypto User	• キーサイン

[1] クラスターサービスは hsm2m.medium でのみ利用できます

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者
- Crypto User (CU)

Syntax

```
aws-cloudhsm > help quorum token-sign generate
Generate a token

Usage: quorum token-sign generate --service <SERVICE> --token <TOKEN>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --service <SERVICE>
          Service the token will be used for

          Possible values:
          - user:
```

```
            User management service is used for executing quorum authenticated user
management operations
          - quorum:
            Quorum management service is used for setting quorum values for any quorum
service
          - cluster:
            Cluster management service is used for executing quorum for cluster
wide configuration managements like mtls enforcement, mtls registration and mtls
deregistration
          - registration:
            Registration service is used for registering a public key for quorum
authentication
          - key-usage:
            Key usage service is used for executing quorum authenticated key usage
operations
          - key-management:
            Key management service is used for executing quorum authenticated key
management operations

      --token <TOKEN>
          Filepath where the unsigned token file will be written
  -h, --help                     Print help
```

例

このコマンドは、クラスター内の HSM ごとに 1 つの未署名トークンを token で指定されたファイルに書き込みます。

Example: クラスターの HSM ごとに 1 つの署名なしトークンを書き込みます

```
aws-cloudhsm > quorum token-sign generate --service user --token /home/tfile
{
  "error_code": 0,
  "data": {
    "filepath": "/home/tfile"
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<SERVICE>

トークンを生成するには、クォーラム承認サービスを指定します。このパラメータは必須です。

有効な値

- ユーザー:クォーラム承認ユーザー管理オペレーションの実行に使用されるユーザー管理サービス。
- クォーラム: すべてのクォーラム承認サービスのクォーラム承認クォーラム値を設定するために使用されるクォーラム管理サービス。
- クラスター: mtls 適用、mtls 登録、mtls 登録解除などのクラスター全体の構成管理のクォーラムを実行するために使用されるクラスター管理サービス。
- 登録:クォーラム認可用のパブリックキーの登録に使用する署名なしトークンを生成します。
- キーの用途: クォーラム認証されたキー使用オペレーションを実行するために使用される、署名なしトークンを生成します。
- キー管理: クォーラム認可されたキー管理オペレーションを実行するために使用される、署名なしトークンを生成します。

必須: はい

<TOKEN>

署名されていないトークンファイルが書き込まれるファイルパス。

必須: はい

関連トピック

- クォーラム認証をサポートするサービス名とタイプ

CloudHSM CLI でクォーラムトークンを一覧表示する

CloudHSM CLI の quorum token-sign list コマンドを使用して、 AWS CloudHSM クラスターに存在するすべてのトークン署名クォーラムトークンを一覧表示します。これには、他のユーザーによって生成されたトークンが含まれます。ただし、トークンはユーザーに紐づけられているため、他のユーザーのトークンが表示されることはありますが、使用できるのは現在ログインしているユーザーに関連付けられているトークンのみです。

サービスのタイプと名前について詳しくは、「クォーラム認証をサポートするサービス名とタイプ」を参照してください。一覧表示されたトークンに関して表示される内容の詳細については、key-management および key-usage サービスに関連付けられたトークンについては「the section called “クォーラム (M-of-N) によるキー管理とキー使用”」を、user、quorum または cluster サービスに関連付けられたトークンについては「the section called “クォーラムによるユーザー管理 (M of N)”」をそれぞれ参照してください。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者
- Crypto User (CU)

Syntax

```
aws-cloudhsm > help quorum token-sign list
List the token-sign tokens in your cluster

Usage: quorum token-sign list

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

例

このコマンドは、 AWS CloudHSM クラスターに存在するすべてのトークン署名トークンを一覧表示します。これには、他のユーザーによって生成されたトークンが含まれます。ただし、トークンはユーザーに紐づけられているため、他のユーザーのトークンが表示されることはありますが、使用できるのは現在ログインしているユーザーに関連付けられているトークンのみです。

Example

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
```

```
    "tokens": [
      {
        "username": "admin",
        "service": "quorum",
        "approvals-required": 2,
        "number-of-approvals": 0,
        "token-timeout-seconds": 397,
        "cluster-coverage": "full"
      },
      {
        "username": "admin",
        "service": "user",
        "approvals-required": 2,
        "number-of-approvals": 0,
        "token-timeout-seconds": 588,
        "cluster-coverage": "full"
      },
      {
        "username": "crypto_user1",
        "service": "key-management",
        "key-reference": "0x00000000002c33f7",
        "minimum-token-count": 1
      },
      {
        "username": "crypto_user1",
        "service": "key-usage",
        "key-reference": "0x00000000002c33f7",
        "minimum-token-count": 1
      }
    ]
  }
}
```


関連トピック

- quorum token-sign generate

CloudHSM CLI でクォーラム値を表示する

CloudHSM CLI の quorum token-sign list-quorum-values コマンドを使用して、 AWS CloudHSM クラスターに設定されたクォーラム値を一覧表示します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- すべてのユーザー。このコマンドは、ログインしていなくても実行できます。

Syntax

```
aws-cloudhsm > help quorum token-sign list-quorum-values
List current quorum values

Usage: quorum token-sign list-quorum-values

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

例

このコマンドは、各サービスの AWS CloudHSM クラスターに設定されたクォーラム値を一覧表示します。

Example

hsm1.medium:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 1
  }
}
```

hsm2m.medium:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
```

```
  "data": {
    "user": 1,
    "quorum": 1,
    "cluster": 1
  }
}
```

関連トピック

- クォーラム認証をサポートするサービス名とタイプ
- mTLS の設定 (推奨)

CloudHSM CLI でクォーラム値を更新する

CloudHSM CLI の quorum token-sign set-quorum-value コマンドを使用して、クォーラム承認サービス用の新しいクォーラム値のトークンを設定します。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- 管理者

Syntax

```
aws-cloudhsm > help quorum token-sign set-quorum-value
Set a quorum value

Usage: quorum token-sign set-quorum-value [OPTIONS] --service <SERVICE> --value <VALUE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --service <SERVICE>
          Service the token will be used for

          Possible values:
          - user:
```

```
            User management service is used for executing quorum authenticated user
 management operations
          - quorum:
            Quorum management service is used for setting quorum values for any quorum
 service
          - cluster:
            Cluster management service is used for executing quorum for cluster
 wide configuration managements like mtls enforcement, mtls registration and mtls
 deregistration

      --value <VALUE>
          Value to set for service

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

  -h, --help
          Print help (see a summary with '-h')
```

例

Example

次の例では、このコマンドは、クラスター内の HSM ごとに 1 つの署名なしトークンを、トークンで指定されたファイルに書き込みます。プロンプトが表示されたら、ファイル内のトークンに署名します。

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
  "data": "Set Quorum Value successful"
}
```

その後、list-quorum-values コマンドを実行して、クォーラム管理サービスのクォーラム値が設定されていることを確認できます。

hsm1.medium:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
```

```
    "quorum": 2
  }
}
```

hsm2m.medium:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 2,
    "cluster": 1
  }
}
```

引数

<CLUSTER_ID>

このオペレーションを実行するクラスターの ID。

必須: 複数のクラスターが設定されている場合。

<APPROVAL>

HSM で承認される署名済みトークンファイルのファイルパス。

<SERVICE>

トークンを生成するには、クォーラム承認サービスを指定します。このパラメータは必須です。サービスのタイプと名前について詳しくは、「クォーラム認証をサポートするサービス名とタイプ」を参照してください。

有効な値

- ユーザー: ユーザー管理サービス。クォーラム承認ユーザー管理オペレーションの実行に使用されるサービス。
- クォーラム: クォーラム管理サービス。あらゆるクォーラム承認サービスのクォーラム承認クォーラム値を設定するために使用されるサービス。
- クラスター: mtls 適用、mtls 登録、mtls 登録解除などのクラスター全体の構成管理のクォーラムを実行するために使用されるクラスター管理サービス。
- 登録: クォーラム認可用のパブリックキーの登録に使用する署名なしトークンを生成します。

必須: はい

<VALUE>

設定するクォーラム値を指定します。最大クォーラム値は 8 です。

必須: はい

関連トピック

- quorum token-sign list-quorum-values
- クォーラム認証をサポートするサービス名とタイプ
- mTLS の設定 (推奨)

AWS CloudHSM 管理ユーティリティ (CMU)

cloudhsm_mgmt_util コマンドラインツールは、Crypto Officer が AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSMs) 内のユーザーを管理するのに役立ちます。 AWS CloudHSM 管理ユーティリティ (CMU) には、ユーザーを作成、削除、一覧表示し、ユーザーパスワードを変更するツールが含まれています。

CMU とキー管理ユーティリティ (KMU) は、クライアント SDK 3 スイートの一部です。クライアント SDK 3 および関連するコマンドラインツール (Key Management Utility および CloudHSM Management Utility) は、HSM タイプが hsm1.medium の場合にのみ利用できます。

また cloudhsm_mgmt_util は、Crypto User (CU) がキーを共有することを可能にするコマンド、およびキー属性を取得して設定することを可能にするコマンドも含まれています。これらのコマンドは、プライマリーキー管理ツールである key_mgmt_util のキー管理コマンドを補完するものです。

クイックスタートについては、「AWS CloudHSM 管理ユーティリティ (CMU) の開始方法」を参照してください。cloudhsm_mgmt_util コマンドの詳細情報とコマンドの使用例については、「AWS CloudHSM 管理ユーティリティコマンドのリファレンス」を参照してください。

トピック

- AWS CloudHSM 管理ユーティリティでサポートされているプラットフォーム
- AWS CloudHSM 管理ユーティリティ (CMU) の開始方法
- CMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)
- CMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)

- AWS CloudHSM 管理ユーティリティコマンドのリファレンス

AWS CloudHSM 管理ユーティリティでサポートされているプラットフォーム

このトピックでは、 AWS CloudHSM 管理ユーティリティ (CMU) がサポートする Linux および Windows プラットフォームについて説明します。

Linux サポート

- Amazon Linux
- Amazon Linux 2
- CentOS 6.10+
- CentOS 7.3+
- CentOS 8
- Red Hat Enterprise Linux (RHEL) 6.10+
- Red Hat Enterprise Linux (RHEL) 7.9+
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 16.04 LTS
- Ubuntu 18.04 LTS

Windows サポート

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

AWS CloudHSM 管理ユーティリティ (CMU) の開始方法

AWS CloudHSM 管理ユーティリティ (CMU) を使用すると、ハードウェアセキュリティモジュール (HSM) ユーザーを管理できます。このトピックを使用して、ユーザーの作成、ユーザーのリスト、CMU のクラスターへの接続など、基本的な HSM ユーザー管理タスクを開始します。

1. CMU を使用するには、まず configure ツールを使用して、クラスタ内の HSM の 1 つから --cmu パラメータと IP アドレスを使用してローカルの CMU 設定を更新する必要があります。CMU を使用するたびにこれを実行して、クラスター内のすべての HSM で HSM ユーザーを管理していることを確認します。

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
   ```

2. 次のコマンドを使用して CLI をインタラクティブモードで起動します。

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:
   \ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

   保有する HSM の数に応じて、出力は次のようになります。

   ```
   Connecting to the server(s), it may take time
   depending on the server(s) load, please wait...

   Connecting to server '10.0.2.9': hostname '10.0.2.9', port 2225...
   Connected to server '10.0.2.9': hostname '10.0.2.9', port 2225.

   Connecting to server '10.0.3.11': hostname '10.0.3.11', port 2225...
   Connected to server '10.0.3.11': hostname '10.0.3.11', port 2225.

   Connecting to server '10.0.1.12': hostname '10.0.1.12', port 2225...
   Connected to server '10.0.1.12': hostname '10.0.1.12', port 2225.
   ```

   key_mgmt_util が実行されていると、プロンプトは aws-cloudhsm> に変わります。

3. loginHSM コマンドを使用して、クラスターにログインします。どのタイプのユーザーでも、このコマンドを使用してクラスターにログインすることができます。

   次のコマンド例では、admin でログインしています。これは、デフォルトの暗号担当者 (CO) です。このユーザーのパスワードは、「クラスターのアクティブ化」を行う場合に設定します。-hpswd パラメータを使用して、パスワードを非表示にします。

   ```
   aws-cloudhsm>loginHSM CO admin -hpswd
   ```

   システムからパスワードの入力を求められます。パスワードを入力するとシステムはパスワードを非表示にし、コマンドが正常に実行されたことと、クラスター上のすべての HSM に接続したことが出力で示されます。

   ```
   Enter password:


   loginHSM success on server 0(10.0.2.9)
   loginHSM success on server 1(10.0.3.11)
   loginHSM success on server 2(10.0.1.12)
   ```

4. listUsers を使用して、クラスター上のすべてのユーザーを一覧表示します。

   ```
   aws-cloudhsm>listUsers
   ```

   CMU は、クラスター上のすべてのユーザーを一覧表示します。

   ```
   Users on server 0(10.0.2.9):
   Number of users found:2

       User Id             User Type       User Name
    MofnPubKey    LoginFailureCnt         2FA
            1               CO              admin                                   NO
                  0               NO
            2               AU              app_user                                NO
                  0               NO
   Users on server 1(10.0.3.11):
   Number of users found:2

       User Id             User Type       User Name
    MofnPubKey    LoginFailureCnt         2FA
   ```

```
        1              CO             admin                          NO
               0              NO
        2              AU             app_user                       NO
               0              NO
Users on server 2(10.0.1.12):
Number of users found:2

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
        1              CO             admin                          NO
               0              NO
        2              AU             app_user                       NO
               0              NO
```

5. createUser を使用して、 **example_user** という CU ユーザーをパスワードは **password1** で作成します。

   アプリケーションで CU ユーザーを使用して、暗号化およびキー管理操作を実行します。ステップ 3 で CO ユーザーとしてログインしたため、CU ユーザーを作成できます。CMU でユーザーの作成および削除や、他のユーザーのパスワード変更などのユーザー管理作業を行うことができるのは、 CO ユーザーのみです。

```
aws-cloudhsm>createUser CU example_user password1
```

   CMU はユーザーの作成操作についてプロンプトを表示します。

```
*************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*************************************************************

Do you want to continue(y/n)?
```

6. CU ユーザーを作成するには **example_user**、**y** と入力します。
7. listUsers を使用して、クラスター上のすべてのユーザーを一覧表示します。

```
aws-cloudhsm>listUsers
```

CMU は先ほど作成した新しい CU ユーザーを含む、クラスター上のすべてのユーザーを一覧表示します。

```
Users on server 0(10.0.2.9):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0               NO
         2              AU              app_user                                NO
               0               NO
         3              CU              example_user                            NO
               0               NO
Users on server 1(10.0.3.11):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0               NO
         2              AU              app_user                                NO
               0               NO
         3              CU              example_user                            NO
               0               NO
Users on server 2(10.0.1.12):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0               NO
         2              AU              app_user                                NO
               0               NO
         3              CU              example_user                            NO
               0               NO
```

8. HSM からログアウトするには、logoutHSM コマンドを使用します。

```
aws-cloudhsm>logoutHSM
```

```
logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
logoutHSM success on server 2(10.0.1.12)
```

9. cloudhsm_mgmt_util を停止するには quit コマンドを使用します。

```
aws-cloudhsm>quit


disconnecting from servers, please wait...
```

CMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)

cloudhsm_mgmt_util (CMU) を使用して AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) を操作するには、Linux 用の AWS CloudHSM クライアントソフトウェアが必要です。このクライアントを以前に作成した Linux Amazon EC2 クライアントインスタンスにインストールする必要があります。Windows を使用している場合は、クライアントをインストールすることもできます。詳細については、「CMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)」を参照してください。

タスク

- ステップ 1. AWS CloudHSM クライアントとコマンドラインツールをインストールする
- ステップ 2. クライアント設定の編集

ステップ 1. AWS CloudHSM クライアントとコマンドラインツールをインストールする

クライアントインスタンスに接続し、次のコマンドを実行して、 AWS CloudHSM クライアントおよびコマンドラインツールをダウンロードしてインストールします。

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-
client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

ステップ 2. クライアント設定の編集

AWS CloudHSM クライアントを使用してクラスターに接続する前に、クライアント設定を編集する必要があります。

クライアント設定を編集するには

1. cloudhsm_mgmt_util にクライアント SDK 3 をインストールする場合は、次の手順を実行して、クラスター内のすべてのノードが同期されていることを確認します。

   a. configure -a ***<IP of one of the HSMs>*** を実行します。

   b. クライアントサービスを再起動します。

   c. configure -m を実行します。

2. 発行証明書 (クラスターの証明書に署名するために使用したもの) を、クライアントインスタンスの次の場所にコピーします：/opt/cloudhsm/etc/customerCA.crt。この場所に証明書をコピーするには、クライアントインスタンスにルートユーザーアクセス権限が必要です。
3. 次の configure コマンドを使用して、 AWS CloudHSM クライアントツールとコマンドラインツールの設定ファイルを更新し、クラスター内の HSM の IP アドレスを指定します。HSM の IP アドレスを取得するには、 AWS CloudHSM コンソールでクラスターを表示するか、 describe-clusters AWS CLI コマンドを実行します。コマンドの出力では、HSM の IP アドレスは EniIp フィールドの値です。複数の HSM がある場合は、いずれかの HSM の IP アドレスを選択してください。どれでも構いません。

```
sudo /opt/cloudhsm/bin/configure -a <IP address>

Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

4. でクラスターをアクティブ化する AWS CloudHSM に移動します。

CMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)

cloudhsm_mgmt_util (CMU) を使用して Windows の AWS CloudHSM クラスターでハードウェアセキュリティモジュール (HSM) を使用するには、Windows 用の AWS CloudHSM クライアントソフトウェアが必要です。このクライアントを以前に作成した Windows Server インスタンスにインストールする必要があります。

> **Note**
>
> - クライアントを更新する場合、以前のインストールに存在する設定ファイルは上書き されません。
> - Windows 用の AWS CloudHSM クライアントインストーラは、Cryptography API: Next Generation (CNG) と Key Storage Provider (KSP) を自動的に登録します。クライアントをアンインストールするには、インストーラーを再度実行し、アンインストール手順に従います。

- Linux を使用している場合は、Linux クライアントをインストールすることもできます。詳細については、「CMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)」を参照してください。

最新 Windows クライアントとコマンドラインツールをインストール (または更新) します。

1. Windows Server インスタンスに接続します。
2. AWSCloudHSMClient-latest.msi インストーラをダウンロードします。
3. cloudhsm_mgmt_util にクライアント SDK 3 をインストールする場合は、次の手順を実行して、クラスター内のすべてのノードが同期されていることを確認します。
   a. configure.exe -a ***<IP of one of the HSMs>*** を実行します。
   b. クライアントサービスを再起動します。
   c. configure.exe -m を実行します。
4. ダウンロード場所に移動して、管理者権限でインストーラー (AWSCloudHSMClient-latest.msi) を実行します。
5. インストーラの手順に従い、インストーラが終了したら 閉じる を選択します。
6. クラスターの証明書に署名するために使用した 自己署名発行証明書を `C:\ProgramData\Amazon\CloudHSM` フォルダにコピーします。
7. 以下のコマンドを実行して、設定ファイルを更新します。更新中は、再設定の間に必ずクライアントを停止してから再開します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM IP address>
```

8. でクラスターをアクティブ化する AWS CloudHSM に移動します。

AWS CloudHSM 管理ユーティリティコマンドのリファレンス

AWS CloudHSM cloudhsm_mgmt_util コマンドラインツールは、Crypto Officer が AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSMs) 内のユーザーを管理するのに役立ちます。また、Crypto User (CU) がキーを共有することを可能にするコマンド、およびキー属性を取得して設定することを可能にするコマンドも含まれています。これらのコマンドは、key_mgmt_util コマンドラインツールのプライマリキー管理コマンドを補完します。

クイックスタートについては、「AWS CloudHSM 管理ユーティリティ (CMU) の開始方法」を参照してください。

cloudhsm_mgmt_util コマンドを実行する前に、 cloudhsm_mgmt_util を起動し、HSM にログインする必要があります。ログインに使用するアカウントのユーザータイプで、使用するコマンドを実行できることを確認してください。

cloudhsm_mgmt_util コマンドをすべて一覧表示するには、次のコマンドを実行します。

```
aws-cloudhsm> help
```

cloudhsm_mgmt_util コマンドの構文を取得するには、次のコマンドを実行します。

```
aws-cloudhsm> help <command-name>
```

> **Note**
>
> 構文は、ドキュメントに従って使用してください。組み込みのソフトウェアヘルプによって追加のオプションが使用できる場合がありますが、これらはサポートされているとみなさず、本番稼働コードで使用しないことをお勧めします。

コマンドを実行するには、コマンド名、または他の cloudhsm_mgmt_util コマンドの名前と区別するのに十分な名前を入力します。

たとえば、HSM 上のユーザーのリストを取得するには、listUsers または listU と入力します。

```
aws-cloudhsm> listUsers
```

cloudhsm_mgmt_util のセッションを終了するには、次のコマンドを実行します。

```
aws-cloudhsm> quit
```

キー属性の解釈については、「AWS CloudHSM KMU のキー属性リファレンス」を参照してください。

以下のトピックでは、cloudhsm_mgmt_util のコマンドについて説明します。

> **Note**
>
> key_mgmt_util と cloudhsm_mgmt_util のコマンドには、同じ名前のものがあります。ただし、コマンドは通常、構文が異なり、出力が異なり、機能がわずかに異なります。

コマンド	説明	ユーザータイプ
changePswd	HSM 上のユーザーのパスワードを変更します。どのユーザーも自分のパスワードを変更できます。CO は誰のパスワードでも変更できます。	CO
createUser	HSM 上のすべてのタイプのユーザーを作成します。	CO
deleteUser	HSM からすべてのタイプのユーザーを削除します。	CO
findAllKeys	ユーザーが所有または共有するキーを取得します。また、各 HSM のすべてのキーの、キー所有権と共有データのハッシュを取得します。	CO、AU
getAttribute	AWS CloudHSM キーの属性値を取得し、ファイルまたは stdout (標準出力) に書き込みます。	CU
getHSMInfo	HSM が実行されているハードウェアに関する情報を取得します。	すべて。ログインは必須ではありません。
getKeyInfo	所有者、共有ユーザー、およびキーのクォーラム認証ステータスを取得します。	すべて。ログインは必須ではありません。

コマンド	説明	ユーザータイプ
info	IP アドレス、ホスト名、ポート、および現在のユーザーを含む、HSM に関する情報を取得します。	すべて。ログインは必須ではありません。
listUsers	各 HSM のユーザー、そのユーザータイプと ID、およびその他の属性を取得します。	すべて。ログインは必須ではありません。
loginHSM および logoutHSM	HSM へのログインとログアウト。	すべて。
quit	cloudhsm_mgmt_util を終了します。	すべて。ログインは必須ではありません。
サーバー	HSM のサーバーモードの起動と終了を行います。	すべて。
RegisterQuorumPubKey	HSM ユーザーを非対称 RSA-2048 キーペアに関連付けます。	CO
setAttribute	既存のキーのラベル、暗号化、復号、ラップ、およびラップ解除の属性の値を変更します。	CU
shareKey	既存のキーを他のユーザーと共有します。	CU
syncKey	クローンされた AWS CloudHSM クラスター間でキーを同期します。	CU、CO
syncUser	クローンされた AWS CloudHSM クラスター間でユーザーを同期します。	CO

CMU を使用してユーザーのパスワードを変更する

AWS CloudHSM cloudhsm_mgmt_util (CMU) の changePswd コマンドを使用して、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) の既存のユーザーのパスワードを変更します。

どのユーザーも自分のパスワードを変更できます。さらに、Crypto Officer（ CO および PCO ）は、別の CO または Crypto User (CU) のパスワードを変更することができます。変更するために現在のパスワードを入力する必要はありません。

> **Note**
>
> 現在 AWS CloudHSM クライアントまたは key_mgmt_util にログインしているユーザーのパスワードを変更することはできません。

changePswd のトラブルシューティングを行うには

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- Crypto Officer (CO)
- Crypto User (CU)

構文

引数は構文の図表で指定された順序で入力します。-hpswd パラメータを使用して、パスワードをマスクします。CO ユーザーの 2 要素認証 (2FA) を有効にするには、-2fa パラメータを入力し、ファイルパスを含めます。詳細については、「the section called “引数”」を参照してください。

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

例

次の例は、changePassword を使用して、現在のユーザーまたは HSM 内の他のユーザーのパスワードをリセットする方法を示しています。

Example: パスワードの変更

HSM のすべてのユーザーは changePswd を使用して自分のパスワードを変更できます。パスワードを変更する前に、info を使用して、ユーザー名やログインユーザーのユーザータイプなど、クラスター内の各 HSM に関する情報を取得します。

次の出力は、Bob が現在 Crypto User (CU) としてログインしていることを示しています。

```
        aws-cloudhsm> info server 0

Id      Name                    Hostname         Port   State           Partition
  LoginState
0       10.1.9.193              10.1.9.193      2225    Connected       hsm-jqici4covtv
  Logged in as 'bob(CU)'

aws-cloudhsm> info server 1

Id      Name                    Hostname         Port   State           Partition
  LoginState
1       10.1.10.7               10.1.10.7       2225    Connected       hsm-ogi3sywxbqx
  Logged in as 'bob(CU)'
```

パスワードを変更するために、Bob は changePswd に続けて、ユーザータイプ、ユーザー名、および新しいパスワードを指定して実行します。

```
aws-cloudhsm> changePswd CU bob newPassword

************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?y
```

```
Changing password for bob(CU) on 2 nodes
```

Example: 別のユーザーのパスワードを変更する

HSM 上の他の CO、または CU のパスワードを変更するには、CO または PCO である必要があります。他のユーザーのパスワードを変更する前に、info コマンドを使用して、自分のユーザタイプが CO または PCO であることを確認してください。

次の出力では、CO である Alice が現在ログインしていることが確認できます。

```
aws-cloudhsm>info server 0

Id      Name             Hostname         Port   State           Partition
 LoginState
0       10.1.9.193       10.1.9.193        2225   Connected      hsm-jqici4covtv
 Logged in as 'alice(CO)'


aws-cloudhsm>info server 1

Id      Name             Hostname         Port   State           Partition
 LoginState
0       10.1.10.7        10.1.10.7        2225   Connected       hsm-ogi3sywxbqx
 Logged in as 'alice(CO)'
```

Alice は別のユーザー、John のパスワードをリセットしようと考えています。パスワードを変更する前に、listUsers コマンドを使用して John のユーザータイプを確認します。

次の出力では、CO ユーザーとして John が表示されています。

```
aws-cloudhsm> listUsers
Users on server 0(10.1.9.193):
Number of users found:5

    User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
        1               PCO             admin                    YES              0
            NO
        2               AU              jane                      NO              0
            NO
```

```
        3               CU              bob                         NO              0
            NO
        4               CU              alice                       NO              0
            NO
        5               CO              john                        NO              0
            NO
Users on server 1(10.1.10.7):
Number of users found:5

    User Id             User Type       User Name           MofnPubKey
 LoginFailureCnt         2FA
        1               PCO             admin                   YES             0
            NO
        2               AU              jane                        NO              0
            NO
        3               CU              bob                         NO              0
            NO
        4               CO              alice                       NO              0
            NO
        5               CO              john                        NO              0
            NO
```

パスワードを変更するために、Alice は changePswd に続けて、John のユーザータイプ、ユーザー名、および新しいパスワードを指定して実行します。

```
aws-cloudhsm>changePswd CO john newPassword

************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?y
Changing password for john(CO) on 2 nodes
```

引数

引数は構文の図表で指定された順序で入力します。-hpswd パラメータを使用して、パスワードをマスクします。CO ユーザーに対して 2FA を有効にするには、-2fa パラメータを入力し、ファイルパスを含めます。2FA での作業の詳細については、「ユーザー 2FA の管理」を参照してください。

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

パスワードを変更しようとしているユーザーの現在のタイプを指定します。changePswd を使用してユーザーのタイプを変更することはできません。

有効な値は、CO、CU、PCO、および PRECO です。

ユーザータイプを取得するには、listUsers を使用します。HSM のユーザータイプの詳細については、「AWS CloudHSM 管理ユーティリティの HSM ユーザータイプ」を参照してください。

必須: はい

<user-name>

ユーザーのわかりやすい名前を指定します。このパラメータは大文字と小文字が区別されません。changePswd を使用してユーザー名を変更することはできません。

必須: はい

<password | -hpswd >

ユーザーの新しいパスワードを指定します。7 ~ 32 文字の文字列を入力します。この値では、大文字と小文字が区別されます。パスワードは、入力するとプレーンテキストで表示されます。パスワードを非表示にするには、-hpswd パラメータをパスワードの代わりに入力し、プロンプトに従います。

必須: はい

[-2fa </path/to/authdata>]

この CO ユーザーに対して 2FA を有効にすることを指定します。2FA の設定に必要なデータを取得するには、-2fa パラメータの後にファイル名でファイルシステム内の場所へのパスを記述します。2FA の操作の詳細については、「ユーザー 2FA の管理」を参照してください。

必須: いいえ

関連トピック

- info
- listUsers

- createUser
- deleteUser

CMU を使用して AWS CloudHSM ユーザーを作成する

cloudhsm_mgmt_util (CMU) の createUser コマンドを使用して、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) にユーザーを作成します。Crypto Officer (CO および PRECO) だけがこのコマンドを実行できます。コマンドが成功すると、クラスター内のすべての HSM にユーザーが作成されます。

createUser のトラブルシューティングを行うには

HSM 設定が不正確な場合、一部の HSM でユーザーが作成されない場合があります。ユーザーが欠落している HSM にユーザーを追加するには、そのユーザーがいない HSM でのみ syncUser コマンドまたは createUser コマンドを使用します。設定エラーを防ぐには、configureツールを -m オプション付きで実行します。

CMU コマンドを実行する前に CMU を起動し、HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto Officer (CO、PRECO)

構文

引数は構文の図表で指定された順序で入力します。-hpswd パラメータを使用して、パスワードをマスクします。2 要素認証 (2FA) を使用して CO ユーザーを作成するには、-2fa パラメータを入力し、ファイルパスを含めます。詳細については、「the section called “引数”」を参照してください。

```
createUser <user-type> <user-name> <password> |-hpswd> [-2fa </path/to/authdata>]
```

例

以下の例では、createUser を使用して HSM に新しいユーザーを作成する方法を示します。

Example: Crypto Officer を作成する

次の例では、クラスター内の HSM に Crypto Officer (CO) を作成します。最初のコマンドは、loginHSM を使用して、Crypto Officer として HSM にログインします。

```
aws-cloudhsm> loginHSM CO admin 735782961

loginHSM success on server 0(10.0.0.1)
loginHSM success on server 1(10.0.0.2)
loginHSM success on server 1(10.0.0.3)
```

2 番目のコマンドでは、createUser コマンドを使用して、HSM 上に新しい Crypto Officer である alice を作成します。

注意メッセージは、コマンドがクラスター内のすべての HSM でユーザーを作成することを説明しています。ただし、コマンドがいずれかの HSM で失敗した場合、その HSM にユーザーは存在しません。続行するには、y と入力します。

出力は、クラスター内の 3 つすべての HSM で新しいユーザーが作成されたことを示しています。

```
aws-cloudhsm> createUser CO alice 391019314

***********************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*************************************************************

Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'

Do you want to continue(y/n)?y
Creating User alice(CO) on 3 nodes
```

コマンドが完了すると、alice には、HSM 上のすべてのユーザーのパスワードを変更するなど、admin CO ユーザーと同じ HSM のアクセス許可が与えられます。

最後のコマンドでは、listUsers コマンドを使用して、クラスターの 3 つの HSM すべてに alice が存在することを検証します。出力は、alice にユーザー ID 3 が割り当てられていることも示しています。.findAllKeys などの他のコマンドでは、このユーザー ID を使用して alice を識別します。

```
aws-cloudhsm> listUsers
```

```
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              alice                                NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              alice                                NO
           0               NO

Users on server 1(10.0.0.3):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              alice                                NO
           0               NO
```

Example: Crypto User を作成する

次の例では、HSM に Crypto User (CU)、bob を作成します。Crypto User はキーを作成して管理することはできますが、ユーザーを管理することはできません。

警告メッセージに「y」と入力して応答すると、クラスター内の 3 つすべての HSM に bob が作成されたことが出力に表示されます。新しい CU は、HSM にログインしてキーを作成および管理できます。

このコマンドでは、パスワード値として defaultPassword を使用しています。その後、bob または他の CO は changePswd コマンドを使用してパスワードを変更できます。

```
aws-cloudhsm> createUser CU bob defaultPassword

************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'

Do you want to continue(y/n)?y
Creating User bob(CU) on 3 nodes
```

引数

引数は構文の図表で指定された順序で入力します。-hpswd パラメータを使用して、パスワードをマスクします。2FA を有効にした CO ユーザーを作成するには、-2fa パラメータを入力し、ファイルパスを含めます。2FA の詳細については、「ユーザー 2FA の管理」を参照してください。

```
createUser <user-type> <user-name> <password> |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

ユーザーのタイプを指定します。このパラメータは必須です。

HSM のユーザータイプの詳細については、「AWS CloudHSM 管理ユーティリティの HSM ユーザータイプ」を参照してください。

有効な値:

- CO: Crypto officers はユーザーを管理できますが、キーを管理することはできません。
- CU: Crypto User は、管理キーを作成し、暗号化オペレーションでキーを使用できます。

HSM のアクティベーション 中にパスワードを割り当てると、PRECO は CO に変換されます。

必須: はい

<user-name>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。

ユーザーの作成後にユーザー名を変更することはできません。cloudhsm_mgmt_util コマンドでは、ユーザータイプとパスワードは大文字と小文字が区別されますが、ユーザー名は区別されません。

必須: はい

<password | -hpswd >

ユーザーのパスワードを指定します。7 ~ 32 文字の文字列を入力します。この値は大文字と小文字が区別されます。パスワードは、入力するとプレーンテキストで表示されます。パスワードを非表示にするには、-hpswd パラメータをパスワードの代わりに入力し、プロンプトに従います。

ユーザーパスワードを変更するには、changePswd を使用します。HSM ユーザーはすべて自分のパスワードを変更できますが、CO ユーザーは HSM のどのタイプのどのユーザーのパスワードでも変更できます。

必須: はい

[-2fa </path/to/authdata>]

2FA が有効な CO ユーザーの作成を指定します。2FA認証の設定に必要なデータを取得するには、-2fa パラメータの後にファイル名でファイルシステム内の場所へのパスを含めます。2FA のセットアップおよび使用の詳細については、「ユーザー 2FA の管理」を参照してください。

必須: いいえ

関連トピック

- listUsers
- deleteUser
- syncUser
- changePswd

CMU を使用して AWS CloudHSM ユーザーを削除する

AWS CloudHSM cloudhsm_mgmt_util (CMU) の deleteUser コマンドを使用して、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) からユーザーを削除します。このコマンドを実行できるのは、Crypto Officer （ CO ）のみです。現在 HSM にログインしているユーザーを削除することはできません。ユーザーの削除の詳細については、「HSM ユーザーを削除する方法」を参照してください。

> **Tip**
>
> キーを所有している Crypto User (CU) を削除することはできません。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- CO

構文

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
deleteUser <user-type> <user-name>
```

例

次の例では、クラスター内の HSM から Crypto Officer (CO) を削除します。最初のコマンドでは、listUsers を使用して HSM のすべてのユーザーを一覧表示します。

出力は、ユーザー 3、alice が HSM の CO であることを示しています。

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                      MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                             YES
          0               NO
```

```
         2              AU              app_user                                  NO
           0               NO
         3              CO              alice                                     NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:3

    User Id             User Type       User Name                           MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                    YES
           0               NO
         2              AU              app_user                                  NO
           0               NO
         3              CO              alice                                     NO
           0               NO

Users on server 1(10.0.0.3):
Number of users found:3

    User Id             User Type       User Name                           MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                                    YES
           0               NO
         2              AU              app_user                                  NO
           0               NO
         3              CO              alice                                     NO
           0               NO
```

2 番目のコマンドでは、deleteUser コマンドを使用して HSM から alice を削除します。

出力は、クラスター内の 3 つすべての HSM でコマンドが成功したことを示しています。

```
aws-cloudhsm> deleteUser CO alice
Deleting user alice(CO) on 3 nodes
deleteUser success on server 0(10.0.0.1)
deleteUser success on server 0(10.0.0.2)
deleteUser success on server 0(10.0.0.3)
```

最後のコマンドでは、listUsers コマンドを使用して alice がクラスターの 3 つの HSM すべてから削除されていることを検証します。

```
aws-cloudhsm> listUsers
```

```
Users on server 0(10.0.0.1):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
Users on server 1(10.0.0.3):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
deleteUser <user-type> <user-name>
```

<user-type>

ユーザーのタイプを指定します。このパラメータは必須です。

> **Tip**
>
> キーを所有している Crypto User (CU) を削除することはできません。

有効な値は CO、CU。

ユーザータイプを取得するには、listUsers を使用します。HSM のユーザータイプの詳細については、「AWS CloudHSM 管理ユーティリティの HSM ユーザータイプ」を参照してください。

必須: はい

<user-name>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。

ユーザーの作成後にユーザー名を変更することはできません。cloudhsm_mgmt_util コマンドでは、ユーザータイプとパスワードは大文字と小文字が区別されますが、ユーザー名は区別されません。

必須: はい

関連トピック

- listUsers
- createUser
- syncUser
- changePswd

CMU を使用して AWS CloudHSM 暗号ユーザーが所有しているキーを一覧表示する

AWS CloudHSM cloudhsm_mgmt_util (CMU) の findAllKeys コマンドを使用して、 の指定された暗号化ユーザー (CU) が AWS CloudHSM 所有または共有するキーを取得します。このコマンドは、各 HSM 上のユーザーデータのハッシュも返します。ハッシュを使用して、ユーザー、キーの所有権、およびキー共有データがクラスター内のすべての HSM で同じかどうかを一目で判断できます。出力では、ユーザーが所有するキーは (o) によって注釈が付けられ、共有キーは (s) によって注釈が付けられます。

HSM のすべての CU が任意のパブリックキーを使用できますが、findAllKeys がパブリックキーを返すのは、指定した CU がそのキーを所有している場合のみです。この動作は、すべての CU ユーザーに公開キーを返す key_mgmt_util の findKey とは異なります。

Crypto Officer (CO および PCO) と Appliance User (AU) のみがこのコマンドを実行できます。Crypto User (CU) は、次のコマンドを実行することができます。

- listUsers – すべてのユーザーを検索する
- key_mgmt_util の findKey で、使用できるキーを見つけます。
- key_mgmt_util の getKeyInfo で、所有または共有している特定のキーの所有者および共有ユーザーを検索します。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- Crypto Officer (CO、PCO)
- Appliance User (AU)

構文

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

例

以下の例では、`findAllKeys` を使用してユーザーのすべてのキーを検索し、HSM のそれぞれのキーユーザー情報のハッシュを取得する方法を示します。

Example: CU のキーを検索する

次の例では、findAllKeys を使用して、ユーザー 4 が所有および共有している HSM のキーを検索します。このコマンドでは、2 番目の引数に値 `0` を使用してハッシュ値を抑制します。オプションのファイル名が省略されるため、コマンドは stdout に書き込みを行います (標準出力)。

出力は、ユーザー 4 が 6 つのキー (8、9、17、262162、19、および 31) を使用できることを示しています。出力では、(s) を使用して、ユーザーが明示的に共有するキーが示されます。ユーザーが

所有するキーは (o) によって示され、ユーザが共有しない対称キーとプライベートキー、およびすべての Crypto User が使用できる公開キーが含まれます。

```
aws-cloudhsm> findAllKeys 4 0
Keys on server 0(10.0.0.1):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 0(10.0.0.1)

Keys on server 1(10.0.0.2):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.2)

Keys on server 1(10.0.0.3):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.3)
```

Example: ユーザーデータが同期されていることを検証する

次の例では findAllKeys を使用して、クラスター内の HSM のすべてが同じユーザー、キーの所有者、およびキー共有の値を含んでいることを検証します。これを行うため、各 HSM のキーユーザーデータのハッシュを取得し、ハッシュ値を比較します。

キーハッシュを取得するため、このコマンドでは 2 番目の引数に値 1 を使用します。オプションのファイル名が省略されるため、コマンドはキーハッシュを stdout に書き込みます。

この例ではユーザー 6 を指定しますが、HSM のキーを所有または共有するどのユーザーに対してもハッシュ値は同じです。指定したユーザーが CO などのキーを所有または共有していない場合、コマンドはハッシュ値を返しません。

出力は、キーハッシュがクラスター内の両方の HSM と同じであることを示しています。HSM の 1 つに異なるユーザー、異なるキー所有者、または異なる共有ユーザーがいた場合、キーハッシュ値は同じにはなりません。

```
aws-cloudhsm> findAllKeys 6 1
Keys on server 0(10.0.0.1):
Number of keys found 3
```

```
number of keys matched from start index 0::3
8(s),9(s),11,17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
Keys on server 1(10.0.0.2):
Number of keys found 3
number of keys matched from start index 0::3
8(s),9(s),11(o),17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

次のコマンドは、ハッシュ値が HSM のすべてのキーのユーザーデータを表すことを示しています。このコマンドでは、ユーザー 3 に対して findAllKeys を使用します。3 つのキーだけを所有または共有しているユーザー 6 とは異なり、ユーザー 3 は 17 個のキーを所有または共有していますが、キーハッシュ値は同じです。

```
aws-cloudhsm> findAllKeys 3 1
Keys on server 0(10.0.0.1):
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
Keys on server 1(10.0.0.2):
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<user id>

指定したユーザーが所有または共有しているすべてのキーを取得します。HSM のユーザーのユーザー ID を入力します。すべてのユーザーのユーザー ID を検索するには、listUsers を使用します。

すべてのユーザー ID が有効ですが、`findAllKeys` は Crypto User (CU) のキーのみを返します。

必須: はい

<key hash>

各 HSM のすべてのキーのユーザーの所有権とデータ共有のハッシュを含める (1) か除外 (0) します。

`user id` 引数がキーを所有または共有しているユーザーを表す場合、キーハッシュが入力されます。異なるキーを所有または共有していても、キーハッシュ値は、HSM でキーを所有または共有しているすべてのユーザーで同じです。ただし、`user id` が CO などのキーを所有または共有していないユーザーを表す場合、ハッシュ値は入力されません。

必須: はい

<output file>

指定したファイルに出力を書き込みます。

必須: いいえ

デフォルト: Stdout

関連トピック

- changePswd
- deleteUser
- listUsers
- syncUser
- key_mgmt_util で findKey

- key_mgmt_util で getKeyInfo

CMU を使用して AWS CloudHSM キー属性値を取得する

AWS CloudHSM cloudhsm_mgmt_util (CMU) の getAttribute コマンドを使用して、 AWS CloudHSM クラスター内のすべてのハードウェアセキュリティモジュール (HSM) からキーの属性値を 1 つ取得し、stdout (標準出力) または ファイルに書き込みます。このコマンドを実行できるのは Crypto User (CU) のみです。

キー属性はキーのプロパティです。キー属性には、キータイプ、クラス、ラベル、ID などの特性と、キーに対して実行できるアクション (暗号化、復号、ラップ、署名、検証など) を表す値が含まれています。

getAttribute は、所有しているキーと共有しているキーに対してのみ使用できます。このコマンド、または、key_mgmt_util の getAttribute コマンドを実行し、キーの属性値の 1 つまたはすべてをファイルに書き込むことができます。

属性とそれを表す定数のリストを取得するには、listAttributes コマンドを使用します。既存のキーの属性値を変更するには、key_mgmt_util の setAttribute および cloudhsm_mgmt_util の setAttribute を使用します。キー属性の解釈については、「AWS CloudHSM KMU のキー属性リファレンス」を参照してください。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- Crypto User (CU)

Syntax

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
getAttribute <key handle> <attribute id> [<filename>]
```

例

次の例では、HSM でキーの抽出可能な属性の値を取得します。次のようなコマンドを使用して HSM からキーをエクスポートできるかどうかを判断できます。

最初のコマンドでは、listAttributes を使用して抽出可能な属性を表す定数を見つけます。出力は、OBJ_ATTR_EXTRACTABLE の定数が 354 であることを示しています。この情報は、「AWS CloudHSM KMU のキー属性リファレンス」の属性と値の説明を使用して検索することもできます。

```
aws-cloudhsm> listAttributes

Following are the possible attribute values for getAttribute:

      OBJ_ATTR_CLASS                = 0
      OBJ_ATTR_TOKEN                = 1
      OBJ_ATTR_PRIVATE              = 2
      OBJ_ATTR_LABEL                = 3
      OBJ_ATTR_TRUSTED              = 134
      OBJ_ATTR_KEY_TYPE             = 256
      OBJ_ATTR_ID                   = 258
      OBJ_ATTR_SENSITIVE            = 259
      OBJ_ATTR_ENCRYPT              = 260
      OBJ_ATTR_DECRYPT              = 261
      OBJ_ATTR_WRAP                 = 262
      OBJ_ATTR_UNWRAP               = 263
      OBJ_ATTR_SIGN                 = 264
      OBJ_ATTR_VERIFY               = 266
      OBJ_ATTR_DERIVE               = 268
      OBJ_ATTR_LOCAL                = 355
      OBJ_ATTR_MODULUS              = 288
      OBJ_ATTR_MODULUS_BITS         = 289
      OBJ_ATTR_PUBLIC_EXPONENT      = 290
      OBJ_ATTR_VALUE_LEN            = 353
      OBJ_ATTR_EXTRACTABLE          = 354
      OBJ_ATTR_NEVER_EXTRACTABLE    = 356
      OBJ_ATTR_ALWAYS_SENSITIVE     = 357
      OBJ_ATTR_DESTROYABLE          = 370
      OBJ_ATTR_KCV                  = 371
      OBJ_ATTR_WRAP_WITH_TRUSTED    = 528
      OBJ_ATTR_WRAP_TEMPLATE        = 1073742353
      OBJ_ATTR_UNWRAP_TEMPLATE      = 1073742354
      OBJ_ATTR_ALL                  = 512
```

2 番目のコマンドは、getAttribute を使用して HSM でキーハンドルが 262170 であるキーの抽出可能な属性の値を取得します。抽出可能な属性を指定するために、コマンドは 354 (属性を表す定数) を使用します。このコマンドではファイル名が指定されないため、getAttribute は出力を stdout に書き込みます。

出力は、HSM のすべてにおいて抽出可能な属性の値が 1 であることを示しています。この値は、キーの所有者がキーをエクスポートできることを示します。値が 0 (0x0) であれば、HSM からキーをエクスポートすることはできません。抽出可能な属性の値は、キーの作成時に設定できますが、変更することはできません。

```
aws-cloudhsm> getAttribute 262170 354

Attribute Value on server 0(10.0.1.10):
OBJ_ATTR_EXTRACTABLE
0x00000001

Attribute Value on server 1(10.0.1.12):
OBJ_ATTR_EXTRACTABLE
0x00000001

Attribute Value on server 2(10.0.1.7):
OBJ_ATTR_EXTRACTABLE
0x00000001
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
getAttribute <key handle> <attribute id> [<filename>]
```

<key-handle>

ターゲットキーのキーハンドルを指定します。各コマンドに指定できるキーは 1 つのみです。キーのキーハンドルを取得するには、key_mgmt_util の findKey を使用します。

指定するキーは所有しているか、共有している必要があります。キーのユーザーを見つけるには、key_mgmt_util の getKeyInfo を使用します。

必須: はい

<attribute id>

属性を識別します。属性を表す定数を入力するか、すべての属性を表す 512 を入力します。たとえば、キーの種類を取得するには「256」と入力します。これは OBJ_ATTR_KEY_TYPE 属性を表す定数です。

属性とその定数のリストアップするには、listAttributes を使用します。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

必須: はい

<filename>

指定したファイルに出力を書き込みます。ファイルパスを入力します。

指定したファイルが既に存在する場合、getAttribute は警告なしにそのファイルを上書きします。

必須: いいえ

デフォルト: Stdout

関連トピック

- key_mgmt_util で getAttribute
- listAttributes
- cloudhsm_mgmt_util で setAttribute
- key_mgmt_util で setAttribute
- キー属性リファレンス

CMU を使用して AWS CloudHSM クラスター内の各 HSM のハードウェア情報を取得する

AWS CloudHSM cloudhsm_mgmt_util (CMU) の getHSMInfo コマンドを使用して、モデル、シリアル番号、FIPS 状態、メモリ、温度、ハードウェアとファームウェアのバージョン番号など、各ハードウェアセキュリティモジュール (HSM) が実行されるハードウェアに関する情報を取得します。この情報には、cloudhsm_mgmt_util が HSM を参照するために使用するサーバー ID も含まれます。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- すべてのユーザー。このコマンドを実行するのに、ログインする必要はありません。

構文

このコマンドにはパラメータはありません。

```
getHSMInfo
```

例

この例では、getHSMInfo を使用して、クラスター内の HSM に関する情報を取得します。

```
aws-cloudhsm> getHSMInfo
Getting HSM Info on 3 nodes
                *** Server 0 HSM Info ***

        Label                :cavium
        Model                :NITROX-III CNN35XX-NFBE

        Serial Number        :3.0A0101-ICM000001
        HSM Flags            :0
        FIPS state           :2 [FIPS mode with single factor authentication]

        Manufacturer ID      :
        Device ID            :10
        Class Code           :100000
        System vendor ID     :177D
        SubSystem ID         :10


        TotalPublicMemory    :560596
        FreePublicMemory     :294568
        TotalPrivateMemory   :0
        FreePrivateMemory    :0
```

```
        Hardware Major       :3
        Hardware Minor       :0

        Firmware Major       :2
        Firmware Minor       :03

        Temperature          :56 C

        Build Number         :13

        Firmware ID          :xxxxxxxxxxxxxxx

...
```

関連トピック

- info

CMU を使用してキーに関する AWS CloudHSM ユーザー情報を取得する

AWS CloudHSM key_mgmt_util (KMU) の getKeyInfo コマンドを使用して、キーを共有している所有者や暗号化ユーザー (CU) など、キーを使用できるユーザーのハードウェアセキュリティモジュール (HSM) ユーザー IDs を返します。キーに対するクォーラム認証が有効になっている場合、getKeyInfo はキーを使用する暗号化オペレーションを承認する必要があるユーザーの数も返します。getKeyInfo は、所有および共有しているキーに対してのみ実行できます。

パブリックキーに対して getKeyInfo を実行すると、HSM のすべてのユーザーがパブリックキーを使用できる場合でも、getKeyInfo はキー所有者のみを返します。HSM のユーザーの HSM ユーザー ID を確認するには、listUsers を使用します。特定のユーザーのキーを見つけるには、key_mgmt_util の findKey -u を使用します。Crypto Officer は cloudhsm_mgmt_util の findAllKeys を使用することができます。

ユーザーは、自分で作成したキーを所有します。自分で作成したキーは、他のユーザーと共有できます。既存のキーを共有または共有解除するには、cloudhsm_mgmt_util の shareKey を使用します。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

構文

```
getKeyInfo -k <key-handle> [<output file>]
```

例

以下の例では、getKeyInfo を使用してキーのユーザーに関する情報を取得する方法を示します。

Example: 非対称キーのユーザーを取得する

次のコマンドでは、キーハンドルが 262162 の AES (非対称) キーを使用できるユーザーを取得します。出力は、キーの所有者がユーザー 3 であり、キーをユーザー 4 および 6 と共有していることを示しています。

ユーザー 3、4、および 6 のみが、キー 262162 に対して getKeyInfo を実行できます。

```
aws-cloudhsm>getKeyInfo 262162
Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 2 user(s):

                4
                6
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 2 user(s):

                4
```

```
                6
```

Example: 対称キーペアのユーザーを取得する

以下のコマンドでは、getKeyInfo を使用して ECC (対称) キーペアのキーを使用できるユーザーを取得します。パブリックキーのキーハンドルは 262179 です。プライベートキーのキーハンドルは 262177 です。

プライベートキー (262177) に対して getKeyInfo を実行すると、キー所有者 (3) とキーを共有している Crypto User (CU) 4 が返されます。

```
aws-cloudhsm>getKeyInfo -k 262177
Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
```

パブリックキー (262179) に対して getKeyInfo を実行すると、キー所有者であるユーザー 3 のみが返されます。

```
aws-cloudhsm>getKeyInfo -k 262179
Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3

Key Info on server 1(10.0.3.6):
```

```
        Token/Flash Key,

        Owned by user 3
```

ユーザー 4 がパブリックキー (および HSM のすべての公開キー) を使用できることを確認するには、key_mgmt_util の findKey の -u パラメータを使用します。

出力は、ユーザー 4 がキーペアのパブリックキー (262179) とプライベートキー (262177) の両方を使用できることを示しています。ユーザー 4 は、他のすべてのパブリックキーと、自分で作成したプライベートキーまたは共有しているプライベートキーを使用することもできます。

```
Command:  findKey -u 4

Total number of keys present 8

 number of keys matched from start index 0::7
11, 12, 262159, 262161, 262162, 19, 20, 21, 262177, 262179

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example: キーのクォーラム認証値 (m_value) を取得する

次の例では、キューの m_value を取得する方法を示します。m_value は、キーを使用する暗号化オペレーションとキーを共有/共有解除するオペレーションを承認するクォーラム内のユーザーの数です。

キーに対してクォーラム認証を有効にすると、ユーザーのクォーラムは、そのキーを使用する暗号化オペレーションを承認する必要があります。クォーラム認証を有効にしてクォーラムサイズを設定するには、キーの作成時に -m_value パラメータを使用します。

次のコマンドでは、genSymKey を使用して 256 ビット AES キーを作成し、ユーザー 4 と共有します。また、m_value パラメータを使用してクォーラム認証を有効にし、クォーラムサイズを 2 ユーザーに設定します。ユーザー数は必要な承認を提供できるだけの大きさが必要です。

出力は、このコマンドでキー 10 が作成されたことを示しています。

```
Command:  genSymKey -t 31 -s 32 -l aes256m2 -u 4 -m_value 2
```

```
        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 10

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

このコマンドは、cloudhsm_mgmt_util の getKeyInfo を使用してキーのユーザー 10 に関する情報を取得します。出力は、キーの所有者がユーザー 3 であり、キーがユーザー 4 と共有されていることを示しています。また、2 ユーザーのクォーラムが、このキーを使用するすべての暗号化オペレーションを承認する必要があることも示しています。

```
aws-cloudhsm>getKeyInfo 10

Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
         2 Users need to approve to use/manage this key
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
         2 Users need to approve to use/manage this key
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
getKeyInfo -k <key-handle> <output file>
```

<key-handle>

HSM で 1 つのキーのキーハンドルを指定します。所有または共有するキーのキーハンドルを入力します。このパラメータは必須です。

必須: はい

<output file>

出力の書き込み先を stdout ではなく、指定したファイルにします。既存のファイルがある場合は、警告なしに上書きされます。

必須: いいえ

デフォルト: stdout

関連トピック

- key_mgmt_util で getKeyInfo
- key_mgmt_util で findKey
- cloudhsm_mgmt_util で findAllKeys
- listUsers
- shareKey

CMU を使用して AWS CloudHSM クラスター内の各 HSM の情報を取得する

AWS CloudHSM cloudhsm_mgmt_util (CMU) の info コマンドを使用して、ホスト名、ポート、IP アドレス、HSM で cloudhsm_mgmt_util にログインしているユーザーの名前とタイプなど、 AWS CloudHSM クラスター内の各ハードウェアセキュリティモジュール (HSM) に関する情報を取得します。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- すべてのユーザー。このコマンドを実行するのに、ログインする必要はありません。

構文

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
info server <server ID>
```

例

この例では、info を使用して、クラスター内の HSM に関する情報を取得します。コマンドは、0 を使用してクラスター内の最初の HSM を参照します。出力は、IP アドレス、ポート、および現在のユーザーのタイプと名前を表示します。

```
aws-cloudhsm> info server 0
Id      Name                    Hostname         Port   State           Partition
        LoginState
0       10.0.0.1                10.0.0.1        2225    Connected       hsm-udw0tkfg1ab
        Logged in as 'testuser(CU)'
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
info server <server ID>
```

<server id>

HSM のサーバー ID を指定します。HSM には、クラスターに追加された順番を示す序数 (0 から始まる) が割り当てられます。HSM のサーバー ID を見つけるには、getHSMInfo を使用します。

必須: はい

関連トピック

- getHSMInfo
- loginHSM および logoutHSM

CMU を使用して AWS CloudHSM キーの属性を一覧表示する

AWS CloudHSM cloudhsm_mgmt_util (CMU) の listAttributes コマンドを使用して、 AWS CloudHSM キーの属性とそれらを表す定数を一覧表示します。これらの定数は、getAttribute コマンドおよび setAttribute コマンドの属性を特定するのに使用します。

キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動 し、HSM に Crypto ユーザー (CU) として ログイン する必要があります。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- すべてのユーザー。このコマンドを実行するのに、ログインする必要はありません。

Syntax

```
listAttributes [-h]
```

例

このコマンドは、key_mgmt_utilで取得および変更できるキー属性と、それらを表す定数を一覧表示します。キー属性の解釈については、「AWS CloudHSM KMU のキー属性リファレンス」を参照してください。すべての属性を表すには、 512 を使用します

```
Command: listAttributes

    Description
===========
The following are all of the possible attribute values for getAttribute.

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
      OBJ_ATTR_TRUSTED                = 134
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ID                     = 258
      OBJ_ATTR_SENSITIVE              = 259
```

```
    OBJ_ATTR_ENCRYPT                = 260
    OBJ_ATTR_DECRYPT                = 261
    OBJ_ATTR_WRAP                   = 262
    OBJ_ATTR_UNWRAP                 = 263
    OBJ_ATTR_SIGN                   = 264
    OBJ_ATTR_VERIFY                 = 266
    OBJ_ATTR_DERIVE                 = 268
    OBJ_ATTR_LOCAL                  = 355
    OBJ_ATTR_MODULUS                = 288
    OBJ_ATTR_MODULUS_BITS           = 289
    OBJ_ATTR_PUBLIC_EXPONENT        = 290
    OBJ_ATTR_VALUE_LEN              = 353
    OBJ_ATTR_EXTRACTABLE            = 354
    OBJ_ATTR_NEVER_EXTRACTABLE      = 356
    OBJ_ATTR_ALWAYS_SENSITIVE       = 357
    OBJ_ATTR_DESTROYABLE            = 370
    OBJ_ATTR_KCV                     = 371
    OBJ_ATTR_WRAP_WITH_TRUSTED       = 528
    OBJ_ATTR_WRAP_TEMPLATE           = 1073742353
    OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
    OBJ_ATTR_ALL                    = 512
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

関連トピック

- getAttribute
- setAttribute
- キー属性リファレンス

CMU を使用するすべての AWS CloudHSM ユーザーを一覧表示する

AWS CloudHSM cloudhsm_mgmt_util の listUsers コマンドを使用して、各ハードウェアセキュリティモジュール (HSM) のユーザーとそのユーザータイプおよびその他の属性を取得します。このコ

マンドは、すべてのユーザータイプで実行できます。このコマンドを実行するのに、ログインする必要はありません。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- すべてのユーザー。このコマンドは、ログインしていなくても実行できます。

構文

このコマンドにはパラメータはありません。

```
listUsers
```

例

このコマンドは、クラスター内の HSM ごとにユーザーとその属性を一覧表示します。User ID 属性を使用して、deleteUser、changePswd、findAllKeys などの他のコマンドでユーザーを特定できます。

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:6

    User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
        1               PCO             admin                   YES              0
            NO
        2               AU              app_user                 NO              0
            NO
        3               CU              crypto_user1             NO              0
            NO
        4               CU              crypto_user2             NO              0
            NO
```

```
         5               CO              officer1                YES             0
             NO
         6               CO              officer2                 NO             0
             NO
Users on server 1(10.0.0.2):
Number of users found:5

    User Id             User Type       User Name           MofnPubKey
 LoginFailureCnt         2FA
         1               PCO             admin                   YES             0
             NO
         2               AU              app_user                 NO             0
             NO
         3               CU              crypto_user1             NO             0
             NO
         4               CU              crypto_user2             NO             0
             NO
         5               CO              officer1                YES             0
             NO
```

この出力が示すユーザー属性は以下のとおりです。

- ユーザー ID: key_mgmt_util および cloudhsm_mgmt_util のコマンドでユーザーを識別します。
- User type: HSM でユーザーが実行できるオペレーションを決定します。
- User Name: ユーザー定義のわかりやすいユーザー名を表示します。
- MofnPubKey: ユーザーがクォーラム認証トークンに署名するためのキーペアを登録済みであるかどうかを示します。
- LoginFailureCnt: ユーザーがログインに失敗した回数を示します。
- 2FA: ユーザーが多要素認証を有効にしていることを示します。

関連トピック

- key_mgmt_util で listUsers
- createUser
- deleteUser
- changePswd

AWS CloudHSM 管理ユーティリティを使用して HSM にログインおよびログアウトする

クラスターの各 HSM でログインおよびログアウトを行うには、 AWS CloudHSM Cloudhsm_mgmt_util の loginHSM と logoutHSM のコマンドを使用します。タイプに関係なく、すべてのユーザーがこのコマンドを使用できます。

> **Note**
>
> ログイン試行回数が 5 回を超えると、アカウントがロックアウトされます。アカウントのロックを解除するには、暗号化オフィサー (CO) が cloudhsm_mgmt_util で changePswd コマンドを使用してパスワードをリセットする必要があります。

loginHSM および logoutHSM のトラブルシューティングを行う場合

これらの cloudhsm_mgmt_util コマンドを実行する前に、cloudhsm_mgmt_util を起動する必要があります。

HSMs を追加または削除する場合は、 AWS CloudHSM クライアントとコマンドラインツールが使用する設定ファイルを更新します。そうしないと、クラスタ内のすべての HSM で変更が有効にならない場合があります。

クラスター内に複数の HSM がある場合は、アカウントがロックアウトされるまでのログイン試行回数の上限が増える可能性があります。これは、CloudHSM クライアントがさまざまな HSM 間で負荷を分散するためです。したがって、ログイン試行は毎回同じ HSM で開始されない場合があります。この機能をテストしている場合は、アクティブな HSM が1つだけのクラスターでテストすることをお勧めします。

2018 年 2 月より前にクラスターを作成した場合、ロックアウトされるまでのログイン試行回数は 20 回です。

ユーザーのタイプ

これらのコマンドは、次のユーザーが実行できます。

- Pre-crypto officer (PRECO)
- Crypto Officer (CO)
- Crypto User (CU)

構文

引数は構文の図表で指定された順序で入力します。-hpswd パラメータを使用して、パスワードをマスクします。2 要素認証 (2FA) でログインするには、-2fa パラメータを入力し、ファイルパスを含めます。詳細については、「the section called “引数”」を参照してください。

```
loginHSM <user-type> <user-name> <password> |-hpswd> [-2fa </path/to/authdata>]
```

```
logoutHSM
```

例

これらの例では、loginHSM および logoutHSM を使用して、クラスターのすべての HSM のログインおよびログアウトを行う方法を示します。

Example: クラスター内の HSM にログインする

このコマンドでは、CO ユーザー admin とパスワード co12345 の認証情報を使用して、クラスターのすべての HSM にログインします。出力には、コマンドが成功し、HSM(この場合は server 0 と server 1) に接続したことが示されています。

```
aws-cloudhsm>loginHSM CO admin co12345

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)
```

Example: 隠しパスワードでログインします

このコマンドは上記の例と同じですが、今回はシステムがパスワードを隠すように指定することを除きます。

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

システムからパスワードの入力を求められます。パスワードを入力すると、システムはパスワードを非表示にし、コマンドが正常に実行されたことと HSM に接続したことが出力 (と) で示されます。

```
Enter password:

loginHSM success on server 0(10.0.2.9)
```

```
loginHSM success on server 1(10.0.3.11)

aws-cloudhsm>
```

Example: HSM からログアウトする

このコマンドは、現在ログインしている HSM (この場合は `server 0` と `server 1`) からログアウトします。出力は、コマンドが成功し、HSM から切断されたことを示します。

```
aws-cloudhsm>logoutHSM

logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
```

引数

引数は構文の図表で指定された順序で入力します。-hpswd パラメータを使用して、パスワードをマスクします。2 要素認証 (2FA) でログインするには、-2fa パラメータを入力し、ファイルパスを含めます。2FA での作業の詳細については、「ユーザー 2FA の管理」を参照してください。

```
loginHSM <user-type> <user-name> <password> |-hpswd> [-2fa </path/to/authdata>]
```

<user type>

HSM にログインしているユーザーのタイプを指定します。詳細については、上記の「ユーザータイプ」を参照してください。

必須: はい

<user name>

HSM にログインしているユーザーのユーザー名を指定します。

必須: はい

<password | -hpswd >

HSM にログインしているユーザーのパスワードを指定します。パスワードを非表示にするには、-hpswd パラメータをパスワードの代わりに入力し、プロンプトに従います。

必須: はい

[-2fa </path/to/authdata>]

この 2FA 対応の CO ユーザーを認証するために、システムが第 2 要素を使用するように指定します。2FA でログインするために必要なデータを取得するには、-2fa パラメータの後にファイル名を指定して、ファイルシステム内の場所へのパスを含めます。2FA の操作の詳細については、「ユーザー 2FA の管理」を参照してください。

必須: いいえ

関連トピック

- cloudhsm_mgmt_util の起動方法
- クラスターのアクティブ化

CMU を使用して AWS CloudHSM ユーザーをキーに関連付ける

AWS CloudHSM cloudhsm_mgmt_util の registerQuorumPubKey コマンドを使用して、ハードウェアセキュリティモジュール (HSM) ユーザーを非対称 RSA-2048 キーペアに関連付けます。HSM ユーザーをキーに関連付けると、それらのユーザーはプライベートキーを使用してクォーラム要求を承認することができ、クラスターは登録された公開キーを使用して、署名がユーザーからのものであることを確認できます。クォーラム認証の詳細については、「クォーラム認証の管理 (M of N アクセス制御)」を参照してください。

> **Tip**
>
> AWS CloudHSM ドキュメントでは、クォーラム認証は M of N (MofN) と呼ばれることがあります。これは、合計 N 人の承認者のうち最小 M 人の承認者を意味します。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto Officer (CO)

構文

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

例

この例では registerQuorumPubKey を使用して、クォーラム認証の要求に対する承認者として Crypto Officer (CO) を登録する方法を示します。このコマンドを実行するには、非対称 RSA-2,048 キーペア、署名付きトークン、および署名なしトークンが必要です。これらの要件の詳細については、「the section called “引数”」を参照してください。

Example: クォーラム認証に HSM ユーザーを登録する

この例では、クォーラム認証の承認者として quorum_officer という CO を登録します。

```
aws-cloudhsm> registerQuorumPubKey CO <quorum_officer> </path/to/quorum_officer.token> </path/to/quorum_officer.token.sig> </path/to/quorum_officer.pub>

*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
registerQuorumPubKey success on server 0(10.0.0.1)
```

最後のコマンドは listUsers コマンドを使用して、quorum_officer が MofN ユーザーとして登録されていることを確認します。

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                           MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                                 NO
          0               NO
        2               AU              app_user                              NO
          0               NO
```

```
        3              CO              quorum_officer                          YES
          0               NO
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

<user-type>

ユーザーのタイプを指定します。このパラメータは必須です。

HSM のユーザータイプの詳細については、「AWS CloudHSM 管理ユーティリティの HSM ユーザータイプ」を参照してください。

有効な値:

- CO: Crypto officers はユーザーを管理できますが、キーを管理することはできません。

必須: はい

<user-name>

ユーザーのわかりやすい名前を指定します。最大長は 31 文字です。許可されている唯一の特殊文字はアンダースコア (_) です。

ユーザーの作成後にユーザー名を変更することはできません。cloudhsm_mgmt_util コマンドでは、ユーザータイプとパスワードは大文字と小文字が区別されますが、ユーザー名は区別されません。

必須: はい

<registration-token>

署名なし登録トークンを含むファイルへのパスを指定します。最大ファイルサイズが 245 バイトの任意のランダムデータを持つことができます。署名なしの登録トークンの作成についての詳細は、「登録トークンの作成と署名」を参照してください。

必須: はい

<signed-registration-token>

登録トークンの SHA256_PKCS メカニズム署名付きハッシュを含むファイルへのパスを指定します。詳細については、「登録トークンの作成と署名」を参照してください。

必須: はい

<public-key>

非対称 RSA-2,048 キーペアの公開キーを含むファイルへのパスを指定します。プライベートキーを使用して、登録トークンに署名します。詳細については、「RSA キーペアの作成」を参照してください。

必須: はい

> **Note**
>
> クラスターは、クォーラム認証と 2 要素認証 (2FA) に同じキーを使用します。つまり、registerQuorumPubKey を使用して 2FA が有効になっているユーザーのクォーラムキーをローテーションすることはできません。キーをローテーションするには、changePswd のようにします。クォーラム認証と 2FA の使用の詳細については、「クォーラム認証と 2FA」を参照してください。

関連トピック

- RSA キーペアの作成
- 登録トークンの作成と署名
- HSM で公開キーを登録する
- クォーラム認証 (M of N アクセス制御) の管理
- クォーラム認証と 2FA
- listUsers

CMU を使用して AWS CloudHSM クラスター内の 1 つの HSM を操作する

AWS CloudHSM cloudhsm_mgmt_util の server コマンドを使用してサーバーモードに入り、特定のハードウェアセキュリティモジュール (HSM) インスタンスと直接やり取りします。

通常、cloudhsm_mgmt_util でコマンドを発行すると、コマンドは指定されたクラスター (グローバルモード) 内のすべての HSM に影響を及ぼします。ただし、単一の HSM にコマンドを発行する必要がある場合があります。たとえば、自動同期に失敗した場合は、クラスター全体の一貫性を維持するために、HSM 上のキーとユーザーを同期する必要がある場合があります。

初期化が成功すると、`aws-cloudhsm >` コマンドプロンプトは、`server >` コマンドプロンプトに置き換わります。

サーバーモードを終了するには、`exit` コマンドを使用します。正常に終了すると、cloudhsm_mgmt_util のコマンドプロンプトに戻ります。

cloudhsm_mgmt_util コマンドを実行する前に、cloudhsm_mgmt_util を起動する必要があります。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- すべてのユーザー。

前提条件

サーバーモードを入力するには、まず送信先 HSM のサーバー数を知る必要があります。サーバー番号は、cloudhsm_mgmt_util が起動時に生成するトレース出力に記載されています。サーバー数は、設定ファイルに表示されている HSM と同じ順序で表示されます。この例では、`server 0` が、目的の HSM に対応するサーバーであることを前提としています。

構文

サーバーモードを起動するには:

```
server <server-number>
```

サーバーモードを終了するには:

```
server> exit
```

例

このコマンドでは、サーバー番号 `0` の HSM のサーバーモードを入力します。

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

サーバーモードを終了するには、exit コマンドを使用します。

```
server0> exit
```

引数

```
server <server-number>
```

<server-number>

送信先 HSM のサーバー数を指定します。

必須: はい

exit コマンドの引数がありません。

関連トピック

- syncKey
- createUser
- deleteUser

CMU を使用して AWS CloudHSM キーの属性を設定する

AWS CloudHSM cloudhsm_mgmt_util の setAttribute コマンドを使用して、HSMs。また、key_mgmt_util の setAttribute コマンドを使用して、セッションキーを永続キーに変換することができます。自分が所有するキーの属性のみ変更できます。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- Crypto User (CU)

Syntax

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
setAttribute <key handle> <attribute id>
```

例

次の例では、対称キーの復号機能を無効にする方法を示します。次のようなコマンドを使用すると、他のキーのラップやラップ解除はできるが、データの暗号化や復号はできないようにラップキーを設定できます。

最初のステップでは、ラップキーを作成します。このコマンドは、key_mgmt_util の genSymKey を使用して、256 ビット AES 対称キーを生成します。出力は、新しいキーのキーハンドルが 14 であることを示しています。

```
$  genSymKey -t 31 -s 32 -l aes256

Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

次に、復号属性の現在の値を確認します。復号属性の属性 ID を取得するには、listAttributes を使用します。出力は、OBJ_ATTR_DECRYPT 属性を表す定数が 261 であることを示しています。キー属性の解釈については、「AWS CloudHSM KMU のキー属性リファレンス」を参照してください。

```
aws-cloudhsm> listAttributes
```

```
Following are the possible attribute values for getAttribute:

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
      OBJ_ATTR_TRUSTED                = 134
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ID                     = 258
      OBJ_ATTR_SENSITIVE              = 259
      OBJ_ATTR_ENCRYPT                = 260
      OBJ_ATTR_DECRYPT                = 261
      OBJ_ATTR_WRAP                   = 262
      OBJ_ATTR_UNWRAP                 = 263
      OBJ_ATTR_SIGN                   = 264
      OBJ_ATTR_VERIFY                 = 266
      OBJ_ATTR_DERIVE                 = 268
      OBJ_ATTR_LOCAL                  = 355
      OBJ_ATTR_MODULUS                = 288
      OBJ_ATTR_MODULUS_BITS           = 289
      OBJ_ATTR_PUBLIC_EXPONENT        = 290
      OBJ_ATTR_VALUE_LEN              = 353
      OBJ_ATTR_EXTRACTABLE            = 354
      OBJ_ATTR_NEVER_EXTRACTABLE      = 356
      OBJ_ATTR_ALWAYS_SENSITIVE       = 357
      OBJ_ATTR_DESTROYABLE            = 370
      OBJ_ATTR_KCV                    = 371
      OBJ_ATTR_WRAP_WITH_TRUSTED      = 528
      OBJ_ATTR_WRAP_TEMPLATE          = 1073742353
      OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
      OBJ_ATTR_ALL                    = 512
```

キー 14 の 復号属性の現在の値を取得するために、次のコマンドは cloudhsm_mgmt_util の getAttribute を使用します。

出力は、復号属性の値がクラスター内の両方の HSM において true (1) であることを示しています。

```
aws-cloudhsm> getAttribute 14 261

Attribute Value on server 0(10.0.0.1):
OBJ_ATTR_DECRYPT
0x00000001
```

```
Attribute Value on server 1(10.0.0.2):
OBJ_ATTR_DECRYPT
0x00000001
```

次のコマンドでは、setAttribute を使用してキー 14 の復号属性の値 (属性 261) を `0` に変更します。これにより、キーの復号機能が無効になります。

出力は、クラスター内の両方の HSM でコマンドが成功したことを示しています。

```
aws-cloudhsm> setAttribute 14 261 0
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)? y
setAttribute success on server 0(10.0.0.1)
setAttribute success on server 1(10.0.0.2)
```

最後のコマンドでも、getAttribute コマンドを使用します。この場合も、キー 14 の復号属性 (属性 261) が取得されます。

今回の出力は、復号属性の値がクラスター内の両方の HSM において false (0) であることを示しています。

```
aws-cloudhsm > getAttribute 14 261
Attribute Value on server 0(10.0.3.6):
OBJ_ATTR_DECRYPT
0x00000000

Attribute Value on server 1(10.0.1.7):
OBJ_ATTR_DECRYPT
0x00000000
```

引数

```
setAttribute <key handle> <attribute idb
```

<key-handle>

所有するキーのキーハンドルを指定します。各コマンドに指定できるキーは 1 つのみです。キーのキーハンドルを取得するには、key_mgmt_util の findKey を使用します。キーのユーザーを確認するには、getKeyInfo を使用します。

必須: はい

<attribute id>

変更する属性を表す定数を指定します。各コマンドに指定できる属性は 1 つのみです。属性とその整数値を取得するには、listAttributes を使用します。キー属性の解釈については、「AWS CloudHSM KMU のキー属性リファレンス」を参照してください。

有効な値:

- 3 – OBJ_ATTR_LABEL。
- 134 – OBJ_ATTR_TRUSTED。
- 260 – OBJ_ATTR_ENCRYPT。
- 261 – OBJ_ATTR_DECRYPT。
- 262 – OBJ_ATTR_WRAP。
- 263 – OBJ_ATTR_UNWRAP。
- 264 – OBJ_ATTR_SIGN。
- 266 – OBJ_ATTR_VERIFY。
- 268 – OBJ_ATTR_DERIVE。
- 370 – OBJ_ATTR_DESTROYABLE。
- 528 – OBJ_ATTR_WRAP_WITH_TRUSTED。
- 1073742353 – OBJ_ATTR_WRAP_TEMPLATE。
- 1073742354 – OBJ_ATTR_UNWRAP_TEMPLATE。

必須: はい

関連トピック

- key_mgmt_util で setAttribute
- getAttribute
- listAttributes

- キー属性リファレンス

CMU を終了する

AWS CloudHSM cloudhsm_mgmt_util の quit コマンドを使用して cloudhsm_mgmt_util を終了します。タイプに関係なく、すべてのユーザーがこのコマンドを使用できます。

cloudhsm_mgmt_util コマンドを実行する前に、cloudhsm_mgmt_util を起動する必要があります。

ユーザーのタイプ

このコマンドは、次のユーザーが実行できます。

- すべてのユーザー。このコマンドは、ログインしていなくても実行できます。

Syntax

```
quit
```

例

このコマンドは、cloudhsm_mgmt_util を終了します。正常に完了すると、通常のコマンドラインに戻ります。このコマンドには出力パラメータはありません。

```
aws-cloudhsm> quit

disconnecting from servers, please wait...
```

関連トピック

- cloudhsm_mgmt_util の起動方法

CMU を使用して AWS CloudHSM キーを共有する

AWS CloudHSM cloudhsm_mgmt_util の shareKey コマンドを使用して、所有しているキーを他の暗号化ユーザーと共有および共有解除します。キーの共有や共有解除ができるのは、キーの所有者のみです。キーの作成時にキーの共有もできます。

キーを共有するユーザーは、暗号化オペレーションでキーを使用することはできますが、そのキーの削除、エクスポート、共有や共有解除、およびキー属性の変更はできません。キーでクォーラム認証

が有効になっている場合、そのクォーラムは、キーの共有や共有解除を行うオペレーションを承認する必要があります。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto User (CU)

Syntax

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

ユーザータイプ: Crypto User (CU)

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

例

以下の例では、shareKey を使用して、所有しているキーを他の Crypto User と共有および共有解除する方法を示します。

Example: キーを共有する

次の例では、shareKey を使用して、現在のユーザーが所有している ECC プライベートキーを HSM の別の Crypto User と共有します。パブリックキーは HSM のすべてのユーザーが利用できるため、共有も共有解除もできません。

最初のコマンドでは、getKeyInfo を使用して、HSM の ECC プライベートキーであるキー 262177 のユーザー情報を取得します。

出力は、キー 262177 の所有者がユーザー 3 であり、他に共有されていないことを示しています。

```
aws-cloudhsm>getKeyInfo 262177
```

```
Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3

Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3
```

このコマンドでは、shareKey を使用して、キー 262177 を HSM の別の Crypto User であるユーザー 4 と共有します。最後の引数では、値 1 で共有オペレーションを指定します。

出力は、クラスター内の両方の HSM でオペレーションが成功したことを示しています。

```
aws-cloudhsm>shareKey 262177 4 1
**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
******************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.3.10)
shareKey success on server 1(10.0.3.6)
```

オペレーションが成功したことを検証するため、この例では最初の getKeyInfo コマンドを再度実行します。

出力は、キー 262177 がユーザー 4 と共有されるようになったことを示しています。

```
aws-cloudhsm>getKeyInfo 262177

Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3
```

```
        also, shared to following 1 user(s):

                4
Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
```

Example: キーを共有解除する

次の例では、対称キーを共有解除します。つまり、キーの共有ユーザーのリストから Crypto User を削除します。

このコマンドでは、shareKey を使用して、キー 6 の共有ユーザーのリストからユーザー 4 を削除します。最後の引数では、値 0 で共有解除オペレーションを指定します。

出力は、両方の HSM でコマンドが成功したことを示しています。その結果、ユーザー 4 は暗号化オペレーションでキー 6 を使用できなくなります。

```
aws-cloudhsm>shareKey 6 4 0
**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
******************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.3.10)
shareKey success on server 1(10.0.3.6)
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<key-handle>

所有するキーのキーハンドルを指定します。各コマンドに指定できるキーは 1 つのみです。キーのキーハンドルを取得するには、key_mgmt_util の findKey を使用します。キーを所有していることを検証するには、getKeyInfo を使用します。

必須: はい

<user id>

キーを共有または共有解除する Crypto User (CU) のユーザー ID を指定します。ユーザーのユーザー ID を検索するには、listUsers を使用します。

必須: はい

<share 1 or unshare 0>

指定したユーザーとキーを共有するには「1」と入力します。キーを共有解除するには、つまり、指定したユーザーをキーの共有ユーザーのリストから削除するには「0」と入力します。

必須: はい

関連トピック

- getKeyInfo

CMU を使用して AWS CloudHSM クラスター全体でキーを同期する

AWS CloudHSM cloudhsm_mgmt_util の syncKey コマンドを使用して、クラスター内の HSM インスタンス間またはクローンされたクラスター間でキーを手動で同期します。通常、クラスター内の HSM インスタンスは自動的にキーを同期させるため、このコマンドを使用する必要はありません。ただし、クローン複製されたクラスター間でキーを同期する場合は、手動で実行する必要があります。クローンされたクラスターは通常、グローバルスケーリングとディザスタリカバリプロセスを簡素化するために、異なる AWS リージョンに作成されます。

syncKey を使用して、任意のクラスター間でキーを同期させることはできません。クラスターの 1 つは、他のクラスターのバックアップから作成する必要があります。さらに、オペレーションが成功するように、両方のクラスターの CO および CU の認証情報が一致している必要があります。詳細については、「HSM ユーザー」を参照してください。

を使用するにはsyncKey、まずソースクラスターから HSM を 1 つ、宛先クラスターから HSM を指定するAWS CloudHSM 設定ファイルを作成する必要があります。これにより、cloudhsm_mgmt_util

は両方の HSM インスタンスに接続することができます。cloudhsm_mgmt_util を起動するには、この設定を使用します。。次に、同期させるキーを所有する CO または CU の認証情報を使用して、ログインします。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto Officer (CO)
- Crypto User (CU)

> **Note**
>
> CO は、任意のキーの syncKey を使用できますが、CU は、所有するキーでのみこのコマンドを使用できます。詳細については、「the section called “ユーザータイプ”」を参照してください。

前提条件

開始する前に、送信先の HSM と同期する送信元 HSM のキーの `key handle` を把握しておく必要があります。`key handle` を検索するには、listUsers コマンドを使用して、名前付きユーザーのすべての識別子を表示します。次に、findAllKeys コマンドを使用して、特定のユーザーに属するすべてのキーを検索します。

また、cloudhsm_mgmt_util が開始時に返すトレース出力に表示される、出典とデスティネーションの HSM に割り当てられた `server IDs` も知っておく必要があります。これらは、設定ファイルに表示されている HSM と同じ順序で表示されます。

「クローンされたクラスター間で CMU を使用する」の手順に従って、新しい設定ファイルで cloudhsm_mgmt_util を初期化します。続いて、server コマンドを発行して、送信元 HSM のサーバーモードを入力します。

構文

> **Note**
>
> syncKey を実行するには、まず HSM のサーバーモードを入力します。これには、同期されるキーが含まれます。

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

ユーザータイプ: Crypto User (CU)

```
syncKey <key handle> <destination hsm>
```

例

server コマンドを実行して、送信元 HSM にログインし、サーバーモードを入力します。この例では、`server 0` が送信元 HSM であると仮定しています。

```
aws-cloudhsm> server 0
```

ここで、syncKey コマンドを実行します。この例では、キー 261251 が、`server 1` に同期されると仮定しています。

```
aws-cloudhsm> syncKey 261251 1
syncKey success
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
syncKey <key handle> <destination hsm>
```

<key handle>

同期させるキーのキーハンドルを指定します。各コマンドに指定できるキーは 1 つのみです。キーのキーハンドルを取得するには、HSM サーバーにログインした状態で findAllKeys を使用します。

必須: はい

<destination hsm>

キーを同期しているサーバーの数を指定します。

必須: はい

関連トピック

- listUsers
- findAllKeys
- の describe-clusters AWS CLI
- サーバー

CMU を使用して AWS CloudHSM クラスター全体でユーザーを同期する

AWS CloudHSM cloudhsm_mgmt_util の syncUser コマンドを使用して、クラスター内の HSM インスタンス間またはクローンされたクラスター間で、Crypto User (CUs) または crypto Officer (COs) を手動で同期します。 AWS CloudHSM はユーザーを自動的に同期しません。通常、クラスター内の HSM がすべてまとめて更新されるように、ユーザーはグローバルモードで管理します。HSM が誤って同期解除された場合 (たとえば、パスワードの変更など)、またはクローンされたクラスター間でユーザーの認証情報を更新する場合は、syncUser を使用する場合があります。クローンされたクラスターは通常、グローバルスケーリングとディザスタリカバリプロセスを簡素化するために、異なる AWS リージョンに作成されます。

CMU コマンドを実行する前に CMU を起動し、 HSM にログインする必要があります。ログインに使用するユーザータイプで、使用するコマンドを実行できることを確認してください。

HSM を追加または削除する場合は、CMU の設定ファイルを更新します。さもないと、クラスター内のすべての HSM で変更が有効にならない場合があります。

ユーザーのタイプ

このコマンドは、次のタイプのユーザーが実行できます。

- Crypto Officer (CO)

前提条件

開始する前に、送信先の HSM と同期する送信元 HSM のユーザーの `user ID` を把握しておく必要があります。`user ID` を確認するには、listUsers コマンドを使用して、クラスター内の HSM のすべてのユーザーを表示します。

また、cloudhsm_mgmt_util が開始時に返すトレース出力に表示される、出典とデスティネーションの HSM に割り当てられた `server ID` も知っておく必要があります。これらは、設定ファイルに表示されている HSM と同じ順序で表示されます。

クローンされたクラスター間で HSM を同期する場合は、「クローンされたクラスター間で CMU を使用する」の説明に従って、cloudhsm_mgmt_util を新しい設定ファイルで初期化します。

syncUser を実行する準備ができたら、server コマンドを発行して、送信元 HSM のサーバーモードを入力します。

構文

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
syncUser <user ID> <server ID>
```

例

server コマンドを実行して、送信元 HSM にログインし、サーバーモードを入力します。この例では、`server 0` が送信元 HSM であると仮定しています。

```
aws-cloudhsm> server 0
```

syncUser コマンドを実行します。この例では、ユーザー 6 が同期対象のユーザー、`server 1` が送信先 HSM であると仮定しています。

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

引数

このコマンドには名前付きパラメータがないため、引数は図表で指定された順序で入力する必要があります。

```
syncUser <user ID> <server ID>
```

<user ID>

同期するユーザーの ID を指定します。各コマンドに指定できるユーザーは 1 つのみです。ユーザーの ID を取得するには、listUsers を使用します。

必須: はい

<server ID>

ユーザーを同期している HSM のサーバーの数を指定します。

必須: はい

関連トピック

- listUsers
- の describe-clusters AWS CLI
- サーバー

AWS CloudHSM キー管理ユーティリティ (KMU)

キー管理ユーティリティ (KMU) は、Crypto User (CU) AWS CloudHSM がハードウェアセキュリティモジュール (HSM) のキーを管理するのに役立つ のコマンドラインツールです。KMU には、キーの生成、削除、インポート、エクスポート、属性の取得と設定、キーの検索、暗号化操作の実行を行う複数のコマンドが含まれています。

KMU と CMU は クライアント SDK 3 スイートの一部です。

クイックスタートについては、「AWS CloudHSM key_mgmt_util の開始方法」を参照してください。コマンドの詳細については、AWS CloudHSM キー管理ユーティリティコマンドのリファレンスを参照してください。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンスを参照してください。

Linux をしている場合に key_mgmt_util を使用するには、クライアントインスタンスに接続し、「KMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)」を参照してください。Windows を使用している場合は、「KMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)」を参照してください。

トピック

- AWS CloudHSM key_mgmt_util の開始方法
- KMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)
- KMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)

- AWS CloudHSM キー管理ユーティリティコマンドのリファレンス

AWS CloudHSM key_mgmt_util の開始方法

AWS CloudHSM には、AWS CloudHSM クライアントソフトウェアに 2 つのコマンドラインツールが含まれています。cloudhsm_mgmt_util ツールには、HSM ユーザーを管理するためのコマンドが含まれています。key_mgmt_util ツールには、キーを管理するためのコマンドが含まれています。key_mgmt_util コマンドラインツールの使用を開始するには、次のトピックを参照してください。

トピック

- AWS CloudHSM key_mgmt_util を設定する
- KMU を使用して AWS CloudHSM クラスター内の HSMs にログインする
- KMU を使用して AWS CloudHSM クラスター内の HSMs からログアウトする
- AWS CloudHSM key_mgmt_util を停止する

コマンドでエラーメッセージまたは予期しない結果が発生した場合は、「トラブルシューティング AWS CloudHSM」のトピックを参照してください。key_mgmt_util コマンドの詳細については、AWS CloudHSM キー管理ユーティリティコマンドのリファレンス を参照してください。

AWS CloudHSM key_mgmt_util を設定する

AWS CloudHSM key_mgmt_util (KMU) を使用する前に、次の設定を完了します。

トピック

- ステップ 1. AWS CloudHSM クライアントを起動する
- ステップ 2. key_mgmt_util の起動

ステップ 1. AWS CloudHSM クライアントを起動する

key_mgmt_util を使用する前に、 AWS CloudHSM クライアントを起動する必要があります。このクライアントは、クラスターの HSM とエンドツーエンドの暗号化された通信を確立するデーモンです。key_mgmt_util ツールはクライアント接続を使用して、クラスターの HSM と通信します。クライアント接続がない場合、key_mgmt_util は動作しません。

AWS CloudHSM クライアントを起動するには

次のコマンドを使用して AWS CloudHSM クライアントを起動します。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合。

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

ステップ 2. key_mgmt_util の起動

AWS CloudHSM クライアントを起動したら、次のコマンドを使用して key_mgmt_util を起動します。

Amazon Linux

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Amazon Linux 2

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 16.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 18.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\key_mgmt_util.exe"
```

key_mgmt_util が実行されていると、プロンプトは Command: に変わります。

Daemon socket connection error メッセージが返されるなど、コマンドが失敗した場合は、設定ファイルを更新してみます。

KMU を使用して AWS CloudHSM クラスター内の HSMs にログインする

key_mgmt_util (KMU) の loginHSM コマンドを使用して、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) にログインします。次のコマンドは、example_user という名前の Crypto User (CU) としてログインします。出力は、クラスターの 3 つすべての HSM のログインが成功したことを示します。

```
Command:  loginHSM -u CU -s example_user -p <PASSWORD>
Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

loginHSM コマンドの構文を次に示します。

```
Command:  loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

KMU を使用して AWS CloudHSM クラスター内の HSMs からログアウトする

key_mgmt_util (KMU) の logoutHSM コマンドを使用して、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) からログアウトします。

```
Command:  logoutHSM
```

```
Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

AWS CloudHSM key_mgmt_util を停止する

exit コマンドを使用して AWS CloudHSM key_mgmt_util を停止します。

```
Command:  exit
```

KMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)

key_mgmt_util (KMU) を使用して AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) を操作するには、Linux 用の AWS CloudHSM クライアントソフトウェアが必要です。このクライアントを以前に作成した Linux EC2 クライアントインスタンスにインストールする必要があります。Windows を使用している場合は、クライアントをインストールすることもできます。詳細については、「KMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)」を参照してください。

タスク

- ステップ 1. AWS CloudHSM クライアントとコマンドラインツールをインストールする
- ステップ 2. クライアント設定の編集

ステップ 1. AWS CloudHSM クライアントとコマンドラインツールをインストールする

クライアントインスタンスに接続し、次のコマンドを実行して、 AWS CloudHSM クライアントおよびコマンドラインツールをダウンロードしてインストールします。

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

ステップ 2. クライアント設定の編集

AWS CloudHSM クライアントを使用してクラスターに接続する前に、クライアント設定を編集する必要があります。

クライアント設定を編集するには

1. 発行証明書 (クラスターの証明書に署名するために使用したもの) を、クライアントインスタンスの次の場所にコピーします：`/opt/cloudhsm/etc/customerCA.crt`。この場所に証明書をコピーするには、クライアントインスタンスにルートユーザーアクセス権限が必要です。
2. 次の configure コマンドを使用して、 AWS CloudHSM クライアントツールとコマンドラインツールの設定ファイルを更新し、クラスター内の HSM の IP アドレスを指定します。HSM の IP アドレスを取得するには、 AWS CloudHSM コンソールでクラスターを表示するか、 describe-

clusters AWS CLI コマンドを実行します。コマンドの出力では、HSM の IP アドレスは EniIp フィールドの値です。複数の HSM がある場合は、いずれかの HSM の IP アドレスを選択してください。どれでも構いません。

```
sudo /opt/cloudhsm/bin/configure -a <IP address>

Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

3. でクラスターをアクティブ化する AWS CloudHSM に移動します。

KMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)

key_mgmt_util (KMU) を使用して Windows 上の AWS CloudHSM クラスターでハードウェアセキュリティモジュール (HSM) を使用するには、Windows 用の AWS CloudHSM クライアントソフトウェアが必要です。このクライアントを以前に作成した Windows Server インスタンスにインストールする必要があります。

最新 Windows クライアントとコマンドラインツールをインストール (または更新) します。

1. Windows Server インスタンスに接続します。
2. ダウンロードページ から最新 (AWSCloudHSMClient-latest.msi) をダウンロードします。
3. ダウンロード場所に移動して、管理者権限でインストーラー (AWSCloudHSMClient-latest.msi) を実行します。
4. インストーラの手順に従い、インストーラが終了したら 閉じる を選択します。
5. クラスターの証明書に署名するために使用した 自己署名発行証明書を C:\ProgramData\Amazon\CloudHSM フォルダにコピーします。
6. 以下のコマンドを実行して、設定ファイルを更新します。更新中は、再設定の間に必ずクライアントを停止してから再開します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM IP address>
```

7. でクラスターをアクティブ化する AWS CloudHSM に移動します。

注意:

- クライアントを更新する場合、以前のインストールに存在する設定ファイルは上書き されません。
- Windows 用の AWS CloudHSM クライアントインストーラは、Cryptography API: Next Generation (CNG) と Key Storage Provider (KSP) を自動的に登録します。クライアントをアンインストールするには、インストーラーを再度実行し、アンインストール手順に従います。
- Linux を使用している場合は、Linux クライアントをインストールすることもできます。詳細については、「KMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)」を参照してください。

AWS CloudHSM キー管理ユーティリティコマンドのリファレンス

key_mgmt_util コマンドラインツールは、キーとその属性の作成、削除、検索など、 AWS CloudHSM クラスター内のハードウェアセキュリティモジュール (HSM) のキーを管理するのに役立ちます。このトピックでは、このツールに含まれている各コマンドについて詳しく説明します。

クイックスタートについては、「AWS CloudHSM key_mgmt_util の開始方法」を参照してください。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。クラスターの HSM とユーザーを管理するコマンドを含む cloudhsm_mgmt_util コマンドラインツールについては、AWS CloudHSM 管理ユーティリティ (CMU) を参照してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

すべての key_mgmt_util コマンドを一覧表示するには、次のように入力します。

```
Command: help
```

特定の key_mgmt_util コマンドのヘルプを表示するには、次のように入力します。

```
Command: <command-name> -h
```

key_mgmt_util セッションを終了するには、次のように入力します。

```
Command: exit
```

次のトピックでは、key_mgmt_util のコマンドについて説明します。

> **Note**
>
> key_mgmt_util と cloudhsm_mgmt_util のコマンドには、同じ名前のものがあります。ただし、コマンドは通常、構文が異なり、出力が異なり、機能がわずかに異なります。

コマンド	説明
aesWrapUnwrap	ファイルのキーの内容を暗号化および復号します。
deleteKey	HSM からキーを削除します。
Error2String	key_mgmt_util 16 進エラーコードに対応するエラーを取得します。
exit	key_mgmt_util を終了します。
exportPrivateKey	プライベートキーのコピーを HSM からディスク上のファイルにエクスポートします。
exportPubKey	パブリックキーのコピーを HSM からファイルにエクスポートします。
exSymKey	対称キーのプレーンテキストコピーを HSM からファイルにエクスポートします。
extractMaskedObject	HSM のキーをマスクされたオブジェクトとして抽出します。
findKey	キーの属性値に基づいてキーを検索します。
findSingleKey	クラスターのすべての HSM にキーが存在することを検証します。
genDSAKeyPair	デジタル署名アルゴリズム (DSA) キーペアを HSM に生成します。

コマンド	説明
genECCKeyPair	楕円曲線暗号 (ECC) キーペアを HSM に生成します。
genRSAKeyPair	RSA 非対称キーペアを HSM に生成します。
genSymKey	対称キーを HSM に生成します。
getAttribute	AWS CloudHSM キーの属性値を取得し、ファイルに書き込みます。
getCaviumPrivKey	プライベートキーのフェイク PEM 形式バージョンを作成し、ファイルにエクスポートします。
getCert	HSM のパーティション証明書を取得し、それをファイルに保存します。
getKeyInfo	キーを使用できるユーザーの HSM ユーザー ID を取得します。 キーがクォーラム制御されている場合は、クォーラムのユーザー数を取得します。
help	key_mgmt_util で使用可能なコマンドに関するヘルプ情報を表示します。
importPrivateKey	プライベートキーを HSM にインポートします。
importPubKey	パブリックキーを HSM にインポートします。
imSymKey	対称キーのプレーンテキストコピーをファイルから HSM 内にインポートします。

コマンド	説明
insertMaskedObject	ディスク上のファイルにあるマスクされたオブジェクトを、そのオブジェクトの元のクラスターの関連クラスターに格納されている HSM に挿入します。関連クラスターとは、元のクラスターのバックアップから生成されたクラスターを指します。
???	あるファイルに、実際のプライベートキーとサンプルの PEM キーのどちらが含まれているかを特定します。
listAttributes	AWS CloudHSM キーの属性とそれを表す定数を一覧表示します。
listUsers	HSM のユーザー、そのユーザータイプと ID、およびその他の属性を取得します。
loginHSM および logoutHSM	クラスターの HSM でログインおよびログアウトを行います。
setAttribute	セッションキーを永続キーに変換します。
sign	選択されたプライベートキーを使ってファイルのための署名を生成します。
unWrapKey	ラップ (暗号化) されたキーをファイルから HSM 内にインポートします。
verify	特定のファイルへの署名に特定のキーが使用されたかどうかを検証します。
wrapKey	キーの暗号化されたコピーを HSM からファイルにエクスポートします。

KMU を使用して AWS CloudHSM ファイルを暗号化および復号する

AWS CloudHSM key_mgmt_util の aesWrapUnwrap コマンドを使用して、ディスク上のファイルの内容を暗号化または復号します。このコマンドは、暗号化キーをラップおよびラップ解除するように設計されていますが、4 KB (4096 バイト) 未満のデータを含むすべてのファイルに使用できます。

aesWrapUnwrap は、PKCS #5 パディングを使用する AES キーラップを使用します。HSM で、ラップまたはラップ解除キーとして、AES キーを使用します。その後、ディスク上の別のファイルに結果が書き込まれます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
aesWrapUnwrap -h

aesWrapUnwrap -m <wrap-unwrap mode>
              -f <file-to-wrap-unwrap>
              -w <wrapping-key-handle>
              [-i <wrapping-IV>]
              [-out <output-file>]
```

例

以下の例では、aesWrapUnwrap を使用してファイルの暗号化キーを暗号化および復号する方法を示します。

Example: 暗号化キーをラップする

次のコマンドでは、aesWrapUnwrap を使用してプレーンテキストで HSM からエクスポートされた Triple DES 対称キーを 3DES.key ファイルにラップします。同様のコマンドを使用して、ファイルに保存されたキーをラップできます。

コマンドは、ラップモードを示す 1 の値で -m パラメータを使用します。-w パラメータを使用して HSM の AES キー (キーハンドル 6) をラップキーとして指定します。ラップしたキーは 3DES.key.wrapped ファイルに書き込まれます。

出力は、コマンドが正常に実行され、推奨されているデフォルトの IV をオペレーションが使用したことを示しています。

```
 Command:  aesWrapUnwrap -f 3DES.key -w 6 -m 1 -out 3DES.key.wrapped

        Warning: IV (-i) is missing.
                 0xA6A6A6A6A6A6A6A6 is considered as default IV
result data:
49 49 E2 D0 11 C1 97 22
17 43 BD E3 4E F4 12 75
8D C1 34 CF 26 10 3A 8D
6D 0A 7B D5 D3 E8 4D C2
79 09 08 61 94 68 51 B7

result written to file 3DES.key.wrapped

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

Example: 暗号化キーをラップ解除する

次の例では、aesWrapUnwrap を使用してラップ (暗号化) したキーをラップ解除 (復号) する方法を示します。HSM にキーをインポートする前に次のようなオペレーションを実行できます。たとえば、暗号化されたキーを imSymKey コマンドでインポートしようとすると、暗号化されたキーには、該当タイプのプレーンテキストキーに必要な形式がないため、エラーが返されます。

コマンドは 3DES.key.wrapped ファイルのキーをラップ解除し、プレーンテキストを 3DES.key.unwrapped ファイルに書き込みます。コマンドは、ラップ解除モードを示す -m の値で 0 パラメータを使用します。-w パラメータを使用して HSM の AES キー (キーハンドル 6) をラップキーとして指定します。ラップしたキーは 3DES.key.unwrapped ファイルに書き込まれます。

```
 Command:  aesWrapUnwrap -m 0 -f 3DES.key.wrapped -w 6 -out 3DES.key.unwrapped

        Warning: IV (-i) is missing.
                 0xA6A6A6A6A6A6A6A6 is considered as default IV
result data:
14 90 D7 AD D6 E4 F5 FA
A1 95 6F 24 89 79 F3 EE
37 21 E6 54 1F 3B 8D 62

result written to file 3DES.key.unwrapped

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-m

モードを指定します。ファイルの内容をラップ (暗号化) するには「1」と入力します。ファイルの内容をラップ解除 (復号) するには「0」と入力します。

必須: はい

-f

ラップするファイルを指定します。4 KB (4096 バイト) 未満のデータを含むファイルを入力します。このオペレーションは、暗号化キーをラップおよびラップ解除するように設計されています。

必須: はい

-w

ラップキーを指定します。HSM で AES キーのキーハンドルを入力します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

ラッピングキーを作成するには、genSymKey を使用して AES キー (タイプ 31) を生成します。

必須: はい

-i

アルゴリズムの代替の初期値 (IV) を指定します。代替を必要とする特殊な条件がなければ、デフォルト値を使用します。

デフォルト: 0xA6A6A6A6A6A6A6A6。デフォルト値は AES キーのラップのアルゴリズム仕様で定義されています。

必須: いいえ

-out

ラップまたはラップ解除されたキーを含む出力ファイルに代わりの名前を指定します。デフォルトは、ローカルディレクトリの wrapped_key (ラップオペレーション) および unwrapped_key (ラップ解除オペレーション) です。

既存のファイルがある場合、aesWrapUnwrap は警告なしに上書きされます。コマンドが失敗すると、aesWrapUnwrap で内容のない出力ファイルが作成されます。

デフォルト: ラップ: `wrapped_key`。ラップ解除: `unwrapped_key`。

必須: いいえ

関連トピック

- exSymKey
- imSymKey
- unWrapKey
- wrapKey

KMU を使用して AWS CloudHSM キーを削除する

key AWS CloudHSM _mgmt_util の deleteKey コマンドを使用して、 AWS CloudHSM クラスターのハードウェアセキュリティモジュール (HSM) からキーを削除します。一度に削除できるキーは 1 つだけです。キーペアの一方のキーを削除しても、ペアの他方のキーには影響がありません。

キーを削除できるのは、キー所有者のみです。キーを共有するユーザーは、キーを暗号化オペレーションで使用することはできますが、削除することはできません。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
deleteKey -h

deleteKey -k
```

例

以下の例では、deleteKey を使用してキーを HSM から削除する方法を示します。

Example: キーを削除する

次のコマンドでは、キーハンドルが 6 のキーを削除します。コマンドが成功すると、deleteKey はクラスターの各 HSM から成功メッセージを返します。

```
Command: deleteKey -k 6

        Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: キーを削除する (失敗)

指定したキーハンドルに対応するキーがないためにコマンドが失敗すると、deleteKey はオブジェクトハンドルが無効であるというエラーメッセージを返します。

```
Command: deleteKey -k 252126

        Cfm3FindKey returned: 0xa8 : HSM Error: Invalid object handle is passed to this
 operation

        Cluster Error Status
        Node id 1 and err state 0x000000a8 : HSM Error: Invalid object handle is passed
 to this operation
        Node id 2 and err state 0x000000a8 : HSM Error: Invalid object handle is passed
 to this operation
```

現在のユーザーがキーの所有者ではないためにコマンドが失敗すると、コマンドはアクセス拒否エラーを返します。

```
Command:  deleteKey -k 262152

Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.
```

パラメータ

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-k

削除するキーのキーハンドルを指定します。HSM のキーのキーハンドルを確認するには、findKey を使用します。

必須: はい

関連トピック

- findKey

KMU を使用して AWS CloudHSM エラーを記述する

AWS CloudHSM key_mgmt_util の Error2String helper コマンドを使用して、key_mgmt_util の 16 進エラーコードに対応するエラーを返します。このコマンドは、コマンドとスクリプトのトラブルシューティングに使用できます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
Error2String -h

Error2String -r <response-code>
```

例

これらの例は、key_mgmt_util エラーコードのエラー文字列を取得するために Error2String を使用する方法を示しています。

Example: エラーの説明を取得する

次のコマンドでは、0xdb エラーコードに関するエラーの説明を取得します。説明では、ユーザーがユーザータイプを間違えたため、key_mgmt_util へのログインに失敗したとなります。key_mgmt_util にログインできるのは Crypto User (CU) のみです。

```
        Command:  Error2String -r 0xdb

        Error Code db maps to HSM Error: Invalid User Type.
```

Example: エラーコードを見つける

この例は、key_mgmt_util エラーのエラーコードの場所を示しています。エラーコード 0xc6 は、文字列 Cfm3<command-name> returned: の後に表示されます。

この例の getKeyInfo では、現在のユーザー (ユーザー 4) が暗号化オペレーションでキーを使用できることを示しています。その場合でも、ユーザーが deleteKey を使用してキーを削除しようとすると、コマンドはエラーコード 0xc6 を返します。

```
        Command:  deleteKey -k 262162

        Cfm3DeleteKey returned: <0xc6> : HSM Error: Key Access is denied

        Cluster Error Status

        Command:  getKeyInfo -k 262162

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 1 user(s):

               4
```

0xc6 エラーが返された場合は、次のような Error2String コマンドを使用してエラーを検索できます。この例で deleteKey コマンドが失敗しているのは、現在のユーザーがキーを共有していても、キーの所有者が別のユーザーであるために、アクセス拒否エラーとなるためです。キーを削除できるのはキーの所有者のみです。

```
        Command:  Error2String -r 0xa8

        Error Code c6 maps to HSM Error: Key Access is denied
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-r

16 進数のエラーコードを指定します。16 進数インジケータ 0x は必須です。

必須: はい

KMU AWS CloudHSM を終了する

AWS CloudHSM key_mgmt_util の exit コマンドを使用して key_mgmt_util を終了します。正常に終了すると、標準のコマンドラインに戻ります。

どの key_mgmt_util コマンドを実行する場合でも、事前に key_mgmt_util を起動する必要があります。

構文

```
exit
```

パラメータ

このコマンドにはパラメータがありません。

関連トピック

- key_mgmt_util の起動

KMU を使用してプライベート AWS CloudHSM キーをエクスポートする

AWS CloudHSM key_mgmt_util の exportPrivateKey コマンドを使用して、ハードウェアセキュリティモジュール (HSM) からファイルに非対称プライベートキーをエクスポートします。HSM では、クリアテキストのキーを直接エクスポートすることはできません。このコマンドは、指定した AES ラップキーを使用してプライベートキーをラップし、ラップされたバイトを復号化して、クリアテキストのプライベートキーをファイルにコピーします。

exportPrivateKey コマンドはキーを HSM から削除したり、キー属性を変更したり、今後の暗号化操作でのキーの使用を禁止したりすることはありません。同じキーを複数回エクスポートできます。

OBJ_ATTR_EXTRACTABLE 属性値が 1 のプライベートキーのみエクスポートすることができます。OBJ_ATTR_WRAP と OBJ_ATTR_DECRYPT 属性値 1 を持つ AES ラップキーを指定する必要があります。キーの属性を確認するには、getAttribute コマンドを使用します。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
exportPrivateKey -h

exportPrivateKey -k <private-key-handle>
                 -w <wrapping-key-handle>
                 -out <key-file>
                 [-m <wrapping-mechanism>]
                 [-wk <wrapping-key-file>]
```

例

この例では、exportPrivateKey を使って HSM からプライベートキーをエクスポートする方法を示します。

Example: プライベートキーをエクスポートする

このコマンドは、ハンドルが 16 のラップキーを使い、ハンドルが 15 のプライベートキーを exportKey.pem という PEM ファイルにエクスポートします。exportPrivateKey は、コマンドが成功すると成功メッセージを返します。

```
Command: exportPrivateKey -k 15 -w 16 -out exportKey.pem

Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS

PEM formatted private key is written to exportKey.pem
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-k

エクスポートするプライベートキーのキーハンドルを指定します。

必須: はい

-w

ラップキーのキーハンドルを指定します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

キーをラップキーとして使用できるかどうかを確認するには、getAttribute を使用して OBJ_ATTR_WRAP 属性 (262) の値を取得します。ラップキーを作成するには、genSymKey を使用して AES キー (タイプ 31) を作成します。

-wk パラメータを使用して外部のラップ解除キーを指定した場合、エクスポート時の (ラップ解除ではなく) ラップに -w ラップキーが使われます。

必須: はい

-out

エクスポートしたプライベートキーの書き込み先とするファイルの名前を指定します。

必須: はい

-m

エクスポートするプライベートキーのラップ方法を指定します。唯一の有効な値は 4 です。これは NIST_AES_WRAP mechanism. を指します。

デフォルト: 4 (NIST_AES_WRAP)

必須: いいえ

-wk

エクスポートするキーをラップ解除するためのキーを指定します。プレーンテキストの AES キーが含まれているファイルのパスと名前を入力します。

このパラメータを含めた場合、exportPrivateKey は、エクスポートするキーをラップする際に -w ファイルのキーを使用し、ラップ解除する際に -wk パラメータで指定されたキーを使用します。

デフォルト: -w パラメータで指定されたラップキーを使用して、ラップとラップ解除の両方を行う。

必須: いいえ

関連トピック

- importPrivateKey
- wrapKey
- unWrapKey
- genSymKey

KMU を使用してパブリック AWS CloudHSM キーをエクスポートする

AWS CloudHSM key_mgmt_util の exportPubKey コマンドを使用して、HSM のパブリックキーをファイルにエクスポートします。これを使用すると、HSM で生成したパブリックキーをエクスポートすることができます。また、このコマンドを使うと、importPubKey コマンドでインポートされたパブリックキーなど、HSM にインポートされたパブリックキーをエクスポートすることもできます。

exportPubKey オペレーションは、キーマテリアルを指定のファイルにコピーします。しかし、キーを HSM から削除したり、キー属性を変更したり、今後の暗号化操作でのキーの使用を禁止したりすることはありません。同じキーを複数回エクスポートできます。

エクスポート可能なパブリックキーは、OBJ_ATTR_EXTRACTABLE 値が 1 であるものに限られます。キーの属性を確認するには、getAttribute コマンドを使用します。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする必要があります。

構文

```
exportPubKey -h

exportPubKey -k <public-key-handle>
             -out <key-file>
```

例

この例では、exportPubKey を使って HSM からパブリックキーをエクスポートする方法を示します。

Example: パブリックキーをエクスポートする

このコマンドは、ハンドルが 10 のパブリックキーを public.pem というファイルにエクスポートします。exportPubKey は、コマンドが成功すると成功メッセージを返します。

```
Command: exportPubKey -k 10 -out public.pem

PEM formatted public key is written to public.pem

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-k

エクスポートするパブリックキーのキーハンドルを指定します。

必須: はい

-out

エクスポートしたパブリックキーの書き込み先とするファイルの名前を指定します。

必須: はい

関連トピック

- importPubKey
- キーの生成

KMU を使用して AWS CloudHSM キーのプレーンテキストコピーをエクスポートする

AWS CloudHSM key_mgmt_util ツールの exSymKey コマンドを使用して、対称キーのプレーンテキストコピーをハードウェアセキュリティモジュール (HSM) からエクスポートし、ディスク上のファイルに保存します。キーの暗号化 (ラップ) されたコピーをエクスポートするには、wrapKey キーを

使用します。プレーンテキストのキー (exSymKey でエクスポートしたものなど) をインポートするには、imSymKey を使用し ます。

エクスポートプロセス中に、exSymKey は、指定した AES キー (ラッピングキー) を使用して、エクスポートするキーをラップ (暗号化) してからラップ解除 (復号化) します。ただし、エクスポートオペレーションの結果は、ディスク上のプレーンテキスト (ラップ解除された) キーとなります。

キーの所有者 (キーを作成した CU ユーザー) のみがキーをエクスポートできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用することはできますが、エクスポートすることはできません。

exSymKey オペレーションは、キーマテリアルをユーザーが指定したファイルにコピーしますが、キーを HSM から削除したり、そのキー属性を変更したり、暗号化オペレーションでのキーの使用を禁止したりはしません。同じキーを複数回エクスポートできます。

exSymKey は対称キーのみをエクスポートします。パブリックキーをエクスポートするには、exportPubKey を使用します。プライベートキーをエクスポートするには、exportPrivateKey を使用します。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
exSymKey -h

exSymKey -k <key-to-export>
         -w <wrapping-key>
         -out <key-file>
         [-m 4]
         [-wk <unwrapping-key-file> ]
```

例

以下の例では、exSymKey を使用してユーザーが所有する対称キーを HSM からエクスポートする方法を示しています。

Example: 3 DES 対称キーをエクスポートする

次のコマンドでは、Triple DES (3DES) 対称キー (キーハンドル 7) をエクスポートします。HSM の既存の AES キー (キーハンドル 6) をラップキーとして使用します。次に、3DES キーのプレーンテキストを 3DES.key ファイルに書き込みます。

出力は、キー 7 (3DES キー) が正常にラップ/ラップ解除されて 3DES.key ファイルに書き込まれたことを示しています。

> **⚠ Warning**
>
> 出力では「ラップされた対称キー」が出力ファイルに書き込まれたことになっていますが、出力ファイルに含まれているのはプレーンテキスト (ラップ解除された) キーです。

```
        Command: exSymKey -k 7 -w 6 -out 3DES.key

       Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "3DES.key"
```

Example: セッション専用のラップキーでエクスポートする

次の例では、セッションでのみ有効なキーをラップキーとして使用する方法を示します。エクスポートするキーはラップされた後で、すぐにラップ解除されて、プレーンテキストとして配信されるため、ラップキーを保持する必要はありません。

以下のコマンドでは、キーハンドル 8 の AES キーを HSM からエクスポートします。このために専用の AES セッションキーを作成して使用します。

最初のコマンドでは、genSymKey を使用して 256 ビット AES キーを作成します。-sess パラメータを使用して、現在のセッションでのみ有効なキーを作成します。

出力は、HSM でキー 262168 が作成されたことを示しています。

```
        Command:  genSymKey -t 31 -s 32 -l AES-wrapping-key -sess

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262168

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

次の例では、キー 8 (エクスポートするキー) が抽出可能な対称キーであることを検証します。また、ラップキー (キー 262168) がセッションでのみ有効な AES キーであることも検証します。findKey コマンドを使用することもできますが、この例では両方のキーの属性をファイルにエクスポートし、grep を使用してファイルの関連する属性値を見つけます。

以下のコマンドでは、getAttribute で -a 値として 512 (すべて) を使用し、キー 8 とキー 262168 のすべての属性を取得します。キー属性の詳細については、「the section called “キー属性リファレンス”」を参照してください。

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

以下のコマンドでは、grep を使用してエクスポートするキー (キー 8) の属性と、セッション専用のラップキー (キー 262168) を検証します。

```
    // Verify that the key to be exported is a symmetric key.
    $  grep -A 1 "OBJ_ATTR_CLASS" attributes/attr_8
    OBJ_ATTR_CLASS
    0x04

    // Verify that the key to be exported is extractable.
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_8
    OBJ_ATTR_EXTRACTABLE
    0x00000001

    // Verify that the wrapping key is an AES key
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_262168
    OBJ_ATTR_KEY_TYPE
    0x1f

    // Verify that the wrapping key is a session key
    $  grep -A 1 "OBJ_ATTR_TOKEN" attributes/attr_262168
    OBJ_ATTR_TOKEN
    0x00

    // Verify that the wrapping key can be used for wrapping
     $  grep -A 1 "OBJ_ATTR_WRAP" attributes/attr_262168
    OBJ_ATTR_WRAP
    0x00000001
```

最後に、exSymKey コマンドを使用してキー 8 をエクスポートします。ラップキーとしてセッションキー (キー 262168) を使用します。

セッションが終了すると、キー 262168 は消滅します。

```
        Command:  exSymKey -k 8 -w 262168 -out aes256_H8.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "aes256_H8.key"
```

Example: 外部のラップ解除キーを使用する

次の例では、外部のラップ解除キーを使用して HSM からキーをエクスポートする方法を示します。

HSM からキーをエクスポートする場合、HSM の AES キーをラップキーとして指定します。デフォルトでは、そのラップキーを使用して、エクスポートするキーがラップおよびラップ解除されます。ただし、-wk パラメータを使用すると、exSymKey でディスク上のファイルにある外部キーを使用してラップ解除できます。この場合は、-w パラメータで指定したキーでターゲットキーをラップし、-wk パラメータで指定したファイルのキーでラップ解除します。

ラップキーは AES (対称) キーである必要があるため、HSM のラップキーとディスク上のラップ解除キーは、キーマテリアルが同じであることが必要です。そのためには、エクスポートオペレーションに先立って、HSM に対するラップキーのインポートまたはエクスポートを行う必要があります。

次の例では、HSM の外部でキーを作成して HSM 内にインポートします。エクスポートする対称キーはキーの内部コピーでラップし、ファイルのキーのコピーでラップ解除します。

最初のコマンドでは、OpenSSL を使用して 256 ビット AES キーを生成します。生成したキーは、aes256-forImport.key ファイルに保存されます。OpenSSL コマンドから返される出力はありませんが、いくつかのコマンドを使用して成功したかどうかを確認できます。次の例では、wc (ワードカウント) ツールを使用してファイルに 32 バイトのデータが保存されていることを確認しています。

```
$  openssl rand -out keys/aes256-forImport.key 32
```

```
$ wc keys/aes256-forImport.key
 0  2 32 keys/aes256-forImport.key
```

次の例では、imSymKey コマンドを使用して aes256-forImport.key ファイルから HSM に AES キーをインポートします。コマンドが完了すると、キーはキーハンドル 262167 で HSM の aes256-forImport.key ファイルに格納されます。

```
Command:  imSymKey -f keys/aes256-forImport.key -t 31 -l aes256-imported -w 6

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 262167

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

次のコマンドでは、エクスポートオペレーションでキーを使用します。このコマンドでは、exSymKey を使用してキー 21 (192 ビット AES キー) をエクスポートします。キーをラップするために、HSM 内にコピーとしてインポートしたキー 262167 を使用します。キーをラップ解除するには、aes256-forImport.key の同じキーマテリアルを使用します。コマンドが完了すると、キー 21 は aes192_h21.key ファイルにエクスポートされます。

```
        Command:  exSymKey -k 21 -w 262167 -out aes192_H21.key -wk aes256-forImport.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

Wrapped Symmetric Key written to file "aes192_H21.key"
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-k

エクスポートするキーのキーハンドルを指定します。このパラメータは必須です。所有する対称キーのキーハンドルを入力します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

キーがエクスポート可能であることを検証するには、getAttribute コマンドを使用して、OBJ_ATTR_EXTRACTABLE 属性の値を取得します。この属性は定数 354 で表されます。また、ユーザーが所有するキーのみをエクスポートすることもできます。キーの所有者を確認するには、getKeyInfo コマンドを使用します。

必須: はい

-w

ラップキーのキーハンドルを指定します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

ラップキーは、エクスポートするキーの暗号化 (ラップ) と復号 (ラップ解除) に使用する HSM のキーです。ラップキーとして使用できるのは AES キーのみです。

任意の AES キー (任意のサイズ) をラップキーとして使用できます。ラップキーは、ターゲットキーをラップし、直後にラップ解除するため、セッション専用の AES キーをラップキーとして使用できます。キーをラップキーとして使用できるかどうかを確認するには、getAttribute を使用して、OBJ_ATTR_WRAP 属性の値を取得します。この属性は定数 262 で表されます。ラップキーを作成するには、genSymKey を使用して AES キー (タイプ 31) を作成します。

-wk パラメータを使用して外部のラップ解除キーを指定すると、エクスポート時に -w ラップキーがラップに使用されます。ただし、ラップ解除には使用されません。

> **Note**
>
> キー 4 は、サポートされていない内部キーを表します。AES キーをラップキーとして作成および管理することをお勧めします。

必須: はい

-out

出力ファイルのパスと名前を指定します。コマンドが成功すると、このファイルに、エクスポートされたキーがプレーンテキストとして配置されます。既存のファイルがある場合は、警告なしに上書きされます。

必須: はい

-m

ラップ方法を指定します。唯一の有効な値は 4 です。これは NIST_AES_WRAP メカニズムを表します。

必須: いいえ

デフォルト: 4

-wk

指定したファイルの AES キーを使用して、エクスポートするキーをラップ解除します。プレーンテキストの AES キーが含まれているファイルのパスと名前を入力します。

このパラメータを含める場合、exSymKey は、-w パラメータで指定した HSM のキーを使用してエクスポートするキーをラップし、-wk ファイルのキーを使用してラップ解除します。-w パラメータと -wk パラメータの値は同じプレーンテキストのキーに解決される必要があります。

必須: いいえ

デフォルト: HSM のラップキーを使用してラップ解除します。

関連トピック

- genSymKey
- imSymKey
- wrapKey

KMU を使用して AWS CloudHSM キーを抽出する

AWS CloudHSM key_mgmt_util の extractMaskedObject コマンドを使用して、ハードウェアセキュリティモジュール (HSM) からキーを抽出し、マスクされたオブジェクトとしてファイルに保存します。マスクされたオブジェクトとは、クローンされたオブジェクトで、insertMaskedObject コマン

ドを使用して再び元のクラスターに挿入して初めて使用可能になります。マスクされたオブジェクトは、生成元であるクラスター、またはそのクラスターのクローンにしか挿入できません。これには、リージョン間でのバックアップのコピーによって生成されたクラスターのクローンバージョンや、そのバックアップを使って新しいクラスターを作成することで生成されたクラスターのクローンバージョンが含まれます。

マスクされたオブジェクトは、抽出不可能なキー (OBJ_ATTR_EXTRACTABLE 値が 0 であるキー) を含め、キーを効率的にオフロードおよび同期する手段です。これにより、 AWS CloudHSM 設定ファイルを更新することなく、異なるリージョンの関連するクラスター間でキーを安全に同期できます。

> **Important**
>
> マスクされたオブジェクトは、挿入時に復号され、元のキーのキーハンドルとは異なるキーハンドルを与えられます。マスクされたオブジェクトには、属性、所有権、共有情報、クォーラム設定など、元のキーに関連付けられているすべてのメタデータが含まれます。アプリケーションのクラスター間でキーを同期する必要がある場合は、代わりに cloudhsm_mgmt_util で syncKey を使用してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、 HSM に ログインする 必要があります。extractMaskedObject コマンドは、キーを所有する CU または任意の CO が使用できます。

Syntax

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
                    -out <object-file>
```

例

この例は、extractMaskedObject を使い、HSM のキーをマスクされたオブジェクトとして抽出する方法を示します。

Example: マスクされたオブジェクトを抽出する

このコマンドは、HSM のハンドルが 524295 であるキーからマスクされたオブジェクトを抽出し、maskedObj というファイルとして保存します。extractMaskedObject は、コマンドが成功すると成功メッセージを返します。

```
Command: extractMaskedObject -o 524295 -out maskedObj

Object was masked and written to file "maskedObj"

        Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-o

マスクされたオブジェクトとして抽出するキーのハンドルを指定します。

必須: はい

-out

マスクされたオブジェクトの保存先とするファイルの名前を指定します。

必須: はい

関連トピック

- insertMaskedObject
- syncKey
- リージョン間のバックアップのコピー
- 以前のバックアップからの AWS CloudHSM クラスターの作成

KMU を使用して属性で AWS CloudHSM キーを検索する

AWS CloudHSM key_mgmt_util の findKey コマンドを使用して、キー属性の値でキーを検索します。設定したすべての基準にキーが一致すると、findKey はキーハンドルを返します。パラメータがない場合、findKey は HSM で使用できるすべてのキーのキーハンドルを返します。特定のキーの属性値を検索するには、getAttribute を使用します。

すべての key_mgmt_util コマンドと同様に、findKey はユーザー固有です。暗号化オペレーションで現在のユーザーが使用できるキーのみが返されます。これには、現在のユーザーが所有しているキーおよび現在のユーザーと共有されているキーが含まれます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
findKey -h

findKey [-c <key class>]
        [-t <key type>]
        [-l <key label>]
        [-id <key ID>]
        [-sess (0 | 1)]
        [-u <user-ids>]
        [-m <modulus>]
        [-kcv <key_check_value>]
```

例

この例では、findKey を使用して HSM でキーを検索および特定する方法を示します。

Example: すべてのキーを検索する

このコマンドは、HSM の現在のユーザーのすべてのキーを検索します。出力には、そのユーザーが所有および共有しているキーと、HSM のすべてのパブリックキーが含まれます。

特定のキーハンドルを持つキーの属性を取得するには、getAttribute を使用します。現在のユーザーが特定のキーを所有しているか共有しているかを判断するには、cloudhsm_mgmt_util で getKeyInfo または findAllKeys を使用します。

```
Command: findKey

Total number of keys present 13

 number of keys matched from start index 0::12
6, 7, 524296, 9, 262154, 262155, 262156, 262157, 262158, 262159, 262160, 262161, 262162

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example: タイプ、ユーザー、およびセッションでキーを検索する

次のコマンドでは、現在のユーザーとユーザー 3 が使用できる永続 AES キーを検索します (ユーザー 3 は、現在のユーザーが表示できない他のキーを使用できる場合があります)。

```
Command: findKey -t 31 -sess 0 -u 3
```

Example: クラスおよびラベルでキーを検索する

次のコマンドでは、2018-sept ラベルで現在のユーザーのすべてのパブリックキーを検索します。

```
Command: findKey -c 2 -l 2018-sept
```

Example: モジュラスで RSA キーを検索する

次のコマンドでは、m4.txt ファイルのモジュラスを使用して作成された、現在のユーザーの RSA キー (タイプ 0) を検索します。

```
Command: findKey -t 0 -m m4.txt
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-t

指定されたタイプのキーを検索します。キークラスを表す定数を入力します。たとえば、3DES キーを検索するには、-t 21 と入力します。

有効な値:

- 0: RSA
- 1: DSA

- 3: EC
- 16: GENERIC_SECRET
- 18: RC4
- 21: Triple DES (3DES)
- 31: AES

必須: いいえ

-c

指定されたクラスのキーを検索します。キークラスを表す定数を入力します。たとえば、パブリックキーを検索するには「`-c 2`」と入力します。

各キータイプに有効な値:

- 2: パブリック。このクラスには、公開 - プライベートのキーペアの公開キーが含まれています。
- 3: プライベート。このクラスには、公開 - プライベートのキーペアの公開キーが含まれています。
- 4: シークレット。このクラスには、対称キーすべてが含まれています。

必須: いいえ

-l

指定されたラベルのキーを検索します。正確なラベルを入力します。`--l` 値にはワイルドカード文字も正規表現も使用できません。

必須: いいえ

-id

指定された ID のキーを検索します。正確な ID 文字列を入力します。`-id` 値にはワイルドカード文字も正規表現も使用できません。

必須: いいえ

-sess

セッションステータスでキーを検索します。現在のセッションでのみ有効なキーを検索するには「`1`」と入力します。永続キーを検索するには「`0`」と入力します。

必須: いいえ

-u

指定されたユーザーと現在のユーザーが共有しているキーを検索します。HSM ユーザー ID のカンマ区切りリスト (-u 3 や -u 4,7 など) を入力します。HSM のユーザーの ID を検索するには、listUsers を使用します。

1 つのユーザー ID を指定すると、findKey はそのユーザーのキーを返します。複数のユーザー ID を指定すると、findKey は指定したユーザーすべてが使用できるキーを返します。

findKey は現在のユーザーが使用できるキーのみを返すため、-u の結果は常に、現在のユーザーのキーと同じかそのサブセットです。任意のユーザーが所有または共有するすべてのキーを取得するために、暗号オフィサー (CO) は cloudhsm_mgmt_util の findAllKeys を使用できます。

必須: いいえ

-m

指定したファイルの RSA モジュラスを使用して作成されたキーを検索します。モジュラスを保存するファイルのパスを入力します。

-m は、一致する RSA モジュラスを含むバイナリファイルを指定します (オプション)。

必須: いいえ

-kcv

指定されたキーのチェック値のキーを検索します。

キーチェック値 (KCV) は、HSM がキーをインポートまたは生成するときに生成されるキーの3バイトのハッシュまたはチェックサムです。キーをエクスポートした後など、HSM の外部で KCV を計算することもできます。次に、KCV 値を比較して、キーのアイデンティティと整合性を確認できます。キーの KCV を取得するには、getAttribute を使用します。

AWS CloudHSM は、次の標準メソッドを使用してキーチェック値を生成します。

- 対称キー: ゼロブロックをキーで暗号化した結果の最初の 3 バイト。
- 非対称キーペア: 公開キーの SHA-1 ハッシュの最初の 3 バイト。
- HMAC キー : 現時点では、HMAC キーの KCV はサポートされていません。

必須: いいえ

Output

findKey の出力には、一致するキーの合計数とそのキーハンドルが一覧表示されます。

```
        Command:  findKey
Total number of keys present 10

 number of keys matched from start index 0::9
6, 7, 8, 9, 10, 11, 262156, 262157, 262158, 262159

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

関連トピック

- findSingleKey
- getKeyInfo
- getAttribute
- cloudhsm_mgmt_util の中の findAllKeys です。
- キー属性リファレンス

KMU を使用して AWS CloudHSM キーを検証する

AWS CloudHSM key_mgmt_util ツールの findSingleKey コマンドを使用して、 AWS CloudHSM クラスター内のすべてのハードウェアセキュリティモジュール (HSM) にキーが存在することを確認します。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
findSingleKey -h

findSingleKey -k <key-handle>
```

例

Example

次のコマンドでは、クラスターの 3 つすべての HSM にキー 252136 が存在することを検証します。

```
Command: findSingleKey -k 252136
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-k

HSM で 1 つのキーのキーハンドルを指定します。このパラメータは必須です。

キーハンドルを見つけるには、findKey コマンドを使用します。

必須: はい

関連トピック

- findKey
- getKeyInfo
- getAttribute

KMU を使用して AWS CloudHSM DSA キーペアを生成する

AWS CloudHSM key_mgmt_util ツールの genDSAKeyPair コマンドを使用して、ハードウェアセキュリティモジュール (HSM) でデジタル署名アルゴリズム (DSA) キーペアを生成します。ユーザーは、モジュラスの長さを指定する必要があります。モジュラスの値はコマンドで生成されます。

ユーザーは、ID を割り当て、他の HSM ユーザーとキーを共有し、抽出不可のキーとセッション終了時に失効するキーを作成することもできます。コマンドが成功すると、キーハンドルが返されます。HSM は、このキーハンドルをパブリックキーとプライベートキーに割り当てます。このキーハンドルでキーを識別することで、他のコマンドでキーを使用できます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し Crypto User (CU) として HSM にログインする 必要があります。

> **Tip**
>
> タイプ、長さ、ラベル、ID など、作成したキーの属性を検索するには、getAttribute を使用します。特定のユーザーのキーを見つけるには、getKeyInfo を使用します。属性値に基づいてキーを検索するには、findKey を使用します。

構文

```
genDSAKeyPair -h

genDSAKeyPair -m <modulus length>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

例

以下の例では、genDSAKeyPair を使用して DSA キーペアを作成する方法を示します。

Example: DSA キーペアを作成する

次のコマンドでは、DSA をラベルとする DSA キーペアを作成します。出力は、パブリックキーのキーハンドルが 19、プライベートキーのキーハンドルが 21 であることを示しています。

```
Command: genDSAKeyPair -m 2048 -l DSA

        Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS
```

```
        Cfm3GenerateKeyPair:    public key handle: 19    private key handle: 21

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: セッション専用の DSA キーペアを作成する

次のコマンドでは、現在のセッションでのみ有効な DSA キーペアを作成します。コマンドは、必須の (一意でない) ラベルに加えて DSA_temp_pair の一意な ID を割り当てます。次のようなキーペアを作成して、セッション専用のトークンの署名および検証ができます。出力は、パブリックキーのキーハンドルが 12、プライベートキーのキーハンドルが 14 であることを示しています。

```
Command: genDSAKeyPair -m 2048 -l DSA-temp -id DSA_temp_pair -sess

        Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

キーペアがセッションでのみ有効であることを確認するには、-sessfindKey の パラメータで、値 1 (true) を使用します。

```
  Command: findKey -sess 1

  Total number of keys present 2

 number of keys matched from start index 0::1
12, 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example: 抽出不可の共有 DSA キーペアを作成する

次のコマンドでは、DSA キーペアを作成します。プライベートキーは、他の 3 ユーザーと共有され、HSM からエクスポートすることはできません。パブリックキーは、すべてのユーザーが使用可能であり、常に抽出可能です。

```
Command:  genDSAKeyPair -m 2048 -l DSA -id DSA_shared_pair -nex -u 3,5,6

Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 11    private key handle: 19

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: クォーラム制御のキーペアを作成する

次のコマンドでは、DSA-mV2 をラベルとする DSA キーペアを作成します。このコマンドでは、-u パラメータを使用してプライベートキーをユーザー 4 および 6 と共有します。-m_value パラメータを使用して、プライベートキーを使用する暗号化オペレーションごとに 2 つ以上の承認のクォーラムを要求します。また、-attest パラメータを使用して、キーペアを生成するファームウェアの整合性を検証します。

出力は、コマンドでキーハンドル 12 のパブリックキーとキーハンドル 17 のプライベートキーが生成され、クラスターファームウェアの認証チェックが合格であることを示しています。

```
Command:  genDSAKeyPair -m 2048 -l DSA-mV2 -m_value 2 -u 4,6 -attest

Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 17

Attestation Check : [PASS]

Cluster Error Status
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

このコマンドでは、プライベートキー (キーハンドル 17) で getKeyInfo を使用します。出力は、キーの所有者が現在のユーザー (ユーザー 3) であり、キーがユーザー 4 および 6 (それ以外はなし) と共有されていることを示しています。また、クォーラム認証が有効になっていて、クォーラムサイズが 2 であることも示しています。

```
        Command:  getKeyInfo -k 17

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3

        also, shared to following 2 user(s):

                4
                6
         2 Users need to approve to use/manage this key
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-m

モジュラスの長さをビット単位で指定します。唯一の有効な値は 2048 です。

必須: はい

-l

キーペアのユーザー定義ラベルを指定します。文字列を入力します。同じラベルがペアの両方のキーに適用されます。label の最大長は 127 文字です。

キーを識別するのに役立つ任意のフレーズを使用できます。ラベルは一意である必要がないため、このラベルを使用してキーをグループ化および分類できます。

必須: はい

-id

キーペアのユーザー定義識別子 (ID) を指定します。クラスター内で一意の文字列を入力します。デフォルトは空の文字列です。指定した ID は、ペアの両方のキーに適用されます。

デフォルト: ID 値なし。

必須: いいえ

-min_srv

`-timeout` パラメーターの値が期限切れになる前に、キーが同期される HSM の最小数を指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、`min_srv` の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-m_value

ペアの秘密キーを使用する暗号化オペレーションを承認する必要があるユーザーの数を指定します。`0` から `8` までの値を入力します。

このパラメータにより、プライベートキーのクォーラム認証要件が確立されます。デフォルト値、`0` で、キーのクォーラム認証機能を無効にします。クォーラム認証が有効になっている場合、指定された数のユーザーがトークンに署名して、プライベートキーを使用する暗号化オペレーション、およびプライベートキーを共有または共有解除するオペレーションを承認する必要があります。

キーの `m_value` を見つけるには、getKeyInfo を使用します。

このパラメーターは、コマンドの `-u` パラメーターが `m_value` 要件を満たすのに十分なユーザーとキーペアを共有している場合にのみ有効です。

デフォルト: 0

必須: いいえ

-nex

プライベートキーを抽出できなくなります。生成されたプライベートキーを HSM からエクスポートする ことはできません。公開キーは常に抽出可能です。

デフォルト: キーペアの公開キーとプライベートキーの両方が抽出可能です。

必須: いいえ

-sess

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、setAttribute を使用します。

デフォルト: キーは永続的です。

必須: いいえ

-timeout

キーが `min_srv` パラメータで指定された HSM の数に同期されるのをコマンドが待機する時間 (秒単位) を指定します。

このパラメータは、`min_srv` パラメータがコマンドでも使用されている場合にのみ有効です。

デフォルト: タイムアウトなし。このコマンドは無期限に待機し、キーが最小数のサーバーと同期されている場合にのみ戻ります。

必須: いいえ

-u

ペアのプライベートキーを指定されたユーザーと共有します。このパラメータは、他の HSM Crypto User (CU) に暗号化オペレーションでプライベートキーを使用する許可を与えます。公開キーは、共有なしですべてのユーザーが使用可能です。

(`-u 5,6`などの) HSM ユーザー ID のカンマ区切りリストを入力します。現在のユーザーの HSM ユーザー ID を含めないでください。HSM で CU の HSM ユーザー ID を検索するには、listUsers を使用します。既存のキーを共有および共有解除するには、cloudhsm_mgmt_util で shareKey を使用します。

デフォルト: 現在のユーザーのみがプライベートキーを使用できます。

必須: いいえ

-attest

クラスターを実行するファームウェアが改ざんされていないことを確認する整合性チェックを実行します。

デフォルト: 認証チェックなし。

必須: いいえ

関連トピック

- genRSAKeyPair
- genSymKey
- genECCKeyPair

KMU を使用して AWS CloudHSM ECC キーペアを生成する

AWS CloudHSM key_mgmt_util ツールの `genECCKeyPair` コマンドを使用して、ハードウェアセキュリティモジュール (HSM) で楕円曲線暗号 (ECC) キーペアを生成します。`genECCKeyPair` コマンドを実行するときは、楕円曲線識別子とキーペアのラベルを指定する必要があります。また、他の CU ユーザーとプライベートキーを共有したり、抽出可能なキー、クォーラム制御キー、およびセッション終了時に失効するキーを作成したりできます。コマンドが成功すると、HSM がパブリックおよびプライベートの ECC キーに割り当てるキーハンドルが返されます。このキーハンドルでキーを識別することで、他のコマンドでキーを使用できます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し Crypto User (CU) として HSM にログインする 必要があります。

> **Tip**
>
> タイプ、長さ、ラベル、ID など、作成したキーの属性を検索するには、getAttribute を使用します。特定のユーザーのキーを見つけるには、getKeyInfo を使用します。属性値に基づいてキーを検索するには、findKey を使用します。

構文

```
genECCKeyPair -h
```

```
genECCKeyPair -i <EC curve id>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

例

以下の例では、genECCKeyPair を使用して ECC キーペアを HSM に作成する方法を示します。

Example: ECC キーペアを作成して検査する

次のコマンドでは、NID_secp384r1 楕円曲線と ecc14 ラベルを使用して ECC キーペアを作成します。出力は、プライベートキーのキーハンドルが 262177、パブリックキーのキーハンドルが 262179 であることを示しています。ラベルは、パブリックキーとプライベートキーの両方に適用されます。

```
Command: genECCKeyPair -i 14 -l ecc14

        Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 262179    private key handle: 262177

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

キーを生成した後、その属性を調べることができます。次のコマンドでは、getAttribute を使用して新しい ECC プライベートキーのすべての属性 (定数 512 で表される) を attr_262177 ファイルに書き込みます。

```
Command: getAttribute -o 262177 -a 512 -out attr_262177
got all attributes of size 529 attr cnt 19
Attributes dumped into attr_262177
```

```
        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

次に、cat コマンドを使用して、attr_262177 属性ファイルの内容を表示します。出力は、キーが楕円曲線プライベートキーであり、このキーは署名には使用できるが、暗号化、復号、ラップ、ラップ解除には使用できないことを示しています。キーは永続的で、エクスポート可能です。

```
$  cat attr_262177

OBJ_ATTR_CLASS
0x03
OBJ_ATTR_KEY_TYPE
0x03
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x00
OBJ_ATTR_DECRYPT
0x00
OBJ_ATTR_WRAP
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x01
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
ecc2
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x0000008a
OBJ_ATTR_KCV
0xbbb32a
OBJ_ATTR_MODULUS
```

```
044a0f9d01d10f7437d9fa20995f0cc742552e5ba16d3d7e9a65a33e20ad3e569e68eb62477a9960a87911e6121d112
OBJ_ATTR_MODULUS_BITS
0x0000019f
```

Example無効な EEC 曲線の使用

次のコマンドでは、NID_X9_62_prime192v1 曲線を使用して ECC キーペアの作成を試行します。この楕円曲線は FIPS モードの HSM に対して無効であるため、コマンドは失敗します。クラスターのサーバーが使用不可であることがメッセージで報告されますが、通常、これはクラスターのサーバーに問題があることを示すものではありません。

```
Command:  genECCKeyPair -i 1 -l ecc1

       Cfm3GenerateKeyPair returned: 0xb3 : HSM Error: This operation violates the
 current configured/FIPS policies

       Cluster Error Status
       Node id 0 and err state 0x30000085 : HSM CLUSTER ERROR: Server in cluster is
 unavailable
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-i

楕円曲線の識別子を指定します。識別子を入力します。

有効な値:

- 2: NID_X9_62_prime256v1
- 14: NID_secp384r1
- 16: NID_secp256k1

必須: はい

-l

キーペアのユーザー定義ラベルを指定します。文字列を入力します。同じラベルがペアの両方のキーに適用されます。label の最大長は 127 文字です。

キーを識別するのに役立つ任意のフレーズを使用できます。ラベルは一意である必要がないため、このラベルを使用してキーをグループ化および分類できます。

必須: はい

-id

キーペアのユーザー定義識別子 (ID) を指定します。クラスター内で一意の文字列を入力します。デフォルトは空の文字列です。指定した ID は、ペアの両方のキーに適用されます。

デフォルト: ID 値なし。

必須: いいえ

-min_srv

`-timeout` パラメーターの値が期限切れになる前に、キーが同期される HSM の最小数を指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、`min_srv` の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-m_value

ペアの秘密キーを使用する暗号化オペレーションを承認する必要があるユーザーの数を指定します。`0` から `8` までの値を入力します。

このパラメータにより、プライベートキーのクォーラム認証要件が確立されます。デフォルト値、`0` で、キーのクォーラム認証機能を無効にします。クォーラム認証が有効になっている場合、指定された数のユーザーがトークンに署名して、プライベートキーを使用する暗号化オペレーション、およびプライベートキーを共有または共有解除するオペレーションを承認する必要があります。

キーの `m_value` を見つけるには、getKeyInfo を使用します。

このパラメーターは、コマンドの `-u` パラメーターが `m_value` 要件を満たすのに十分なユーザーとキーペアを共有している場合にのみ有効です。

デフォルト: 0

必須: いいえ

-nex

プライベートキーを抽出できなくなります。生成されたプライベートキーを HSM からエクスポートする ことはできません。公開キーは常に抽出可能です。

デフォルト: キーペアの公開キーとプライベートキーの両方が抽出可能です。

必須: いいえ

-sess

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、setAttribute を使用します。

デフォルト: キーは永続的です。

必須: いいえ

-timeout

キーが `min_srv` パラメータで指定された HSM の数に同期されるのをコマンドが待機する時間 (秒単位) を指定します。

このパラメータは、`min_srv` パラメータがコマンドでも使用されている場合にのみ有効です。

デフォルト: タイムアウトなし。このコマンドは無期限に待機し、キーが最小数のサーバーと同期されている場合にのみ戻ります。

必須: いいえ

-u

ペアのプライベートキーを指定されたユーザーと共有します。このパラメータは、他の HSM Crypto User (CU) に暗号化オペレーションでプライベートキーを使用する許可を与えます。公開キーは、共有なしですべてのユーザーが使用可能です。

(-u 5,6などの) HSM ユーザー ID のカンマ区切りリストを入力します。現在のユーザーの HSM ユーザー ID を含めないでください。HSM で CU の HSM ユーザー ID を検索するには、listUsers を使用します。既存のキーを共有および共有解除するには、cloudhsm_mgmt_util で shareKey を使用します。

デフォルト: 現在のユーザーのみがプライベートキーを使用できます。

必須: いいえ

-attest

クラスターを実行するファームウェアが改ざんされていないことを確認する整合性チェックを実行します。

デフォルト: 認証チェックなし。

必須: いいえ

関連トピック

- genSymKey
- genRSAKeyPair
- genDSAKeyPair

KMU を使用して AWS CloudHSM RSA キーペアを生成する

AWS CloudHSM key_mgmt_util ツールの genRSAKeyPair コマンドを使用して、RSA 非対称キーペアを生成します。ユーザーは、キーのタイプ、モジュラスの長さ、および公開指数を指定します。コマンドは、指定した長さのモジュラスを生成し、キーペアを作成します。ユーザーは、ID を割り当て、他の HSM ユーザーとプライベートキーを共有し、抽出不可のキーとセッション終了時に失効するキーを作成できます。コマンドが成功すると、HSM がキーに割り当てるキーハンドルが返されます。このキーハンドルでキーを識別して他のコマンドで使用できます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

> **Tip**
>
> タイプ、長さ、ラベル、ID など、作成したキーの属性を検索するには、getAttribute を使用します。特定のユーザーのキーを見つけるには、getKeyInfo を使用します。属性値に基づいてキーを検索するには、findKey を使用します。

構文

```
genRSAKeyPair -h

genRSAKeyPair -m <modulus length>
              -e <public exponent>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

例

以下の例では、genRSAKeyPair を使用して非対称キーペアを HSM に作成する方法を示します。

Example: RSA キーペアを作成して検査する

このコマンドでは、モジュラス 2048 ビットで指数が 65537 の RSA キーペアを作成します。出力は、パブリックキーのハンドルが 2100177、プライベートキーのハンドルが 2100426 であることを示しています。

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa_test

Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 2100177    private key handle:
 2100426

        Cluster Status:
        Node id 0 status: 0x00000000 : HSM Return: SUCCESS
```

```
        Node id 1 status: 0x00000000 : HSM Return: SUCCESS
```

次のコマンドでは、getAttribute を使用して、先ほど作成したパブリックキーの属性を取得します。出力の書き込み先は attr_2100177 ファイルです。この属性ファイルの内容を取得する cat コマンドが続けて実行されます。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

結果の 16 進値は、RSA タイプ (OBJ_ATTR_CLASS 0x02) のパブリックキー (OBJ_ATTR_KEY_TYPE 0x00) であることを示しています。この公開キーを使用して暗号化 (OBJ_ATTR_ENCRYPT 0x01) はできますが、復号 (OBJ_ATTR_DECRYPT 0x00) を行うことはできません。結果には、キーの長さ (512、0x200)、モジュラス、モジュラスの長さ (2048、0x800)、およびパブリック指数 (65537、0x10001) も含まれています。

```
Command:  getAttribute -o 2100177 -a 512 -out attr_2100177

Attribute size: 801, count: 26
Written to: attr_2100177 file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS

$  cat attr_2100177
OBJ_ATTR_CLASS
0x02
OBJ_ATTR_KEY_TYPE
0x00
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x00
OBJ_ATTR_WRAP
0x01
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x01
OBJ_ATTR_LOCAL
```

```
0x01
OBJ_ATTR_SENSITIVE
0x00
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
rsa_test
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x00000200
OBJ_ATTR_KCV
0xc51c18
OBJ_ATTR_MODULUS
0xbb9301cc362c1d9724eb93da8adab0364296bde7124a241087d9436b9be57e4f7780040df03c2c
1c0fe6e3b61aa83c205280119452868f66541bbbffacbbe787b8284fc81deaeef2b8ec0ba25a077d
6983c77a1de7b17cbe8e15b203868704c6452c2810344a7f2736012424cf0703cf15a37183a1d2d0
97240829f8f90b063dd3a41171402b162578d581980976653935431da0c1260bfe756d85dca63857
d9f27a541676cb9c7def0ef6a2a89c9b9304bcac16fdf8183c0a555421f9ad5dfeb534cf26b65873
970cdf1a07484f1c128b53e10209cc6f7ac308669112968c81a5de408e7f644fe58b1a9ae1286fec
b3e4203294a96fae06f8f0db7982cb5d7f
OBJ_ATTR_MODULUS_BITS
0x00000800
OBJ_ATTR_PUBLIC_EXPONENT
0x010001
OBJ_ATTR_TRUSTED
0x00
OBJ_ATTR_WRAP_WITH_TRUSTED
0x00
OBJ_ATTR_DESTROYABLE
0x01
OBJ_ATTR_DERIVE
0x00
OBJ_ATTR_ALWAYS_SENSITIVE
0x00
OBJ_ATTR_NEVER_EXTRACTABLE
0x00
```

Example: 共有 RSA キーペアを生成する

次のコマンドでは、RSA キーペアを生成し、HSM の別の CU であるユーザー 4 とプライベートキーを共有します。コマンドでは、m_value パラメータを使用して少なくとも 2 つの承認を要求した上で、ペアのプライベートキーを暗号化オペレーションで使用できるようにします。m_value パラ

メータを使用する場合、コマンドで -u も使用する必要があります。これにより、m_value がユーザーの合計数 (-u の数値 + 所有者) を超えないようにします。

```
Command:  genRSAKeyPair -m 2048 -e 65537 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2

       Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

       Cfm3GenerateKeyPair:    public key handle: 27    private key handle: 28

       Cluster Error Status
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
       Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-m

モジュラスの長さをビット単位で指定します。最小値は 2,048 です。

必須: はい

-e

パブリック指数を指定します。値は、65537 以上の奇数にする必要があります

必須: はい

-l

キーペアのユーザー定義ラベルを指定します。文字列を入力します。同じラベルがペアの両方のキーに適用されます。label の最大長は 127 文字です。

キーを識別するのに役立つ任意のフレーズを使用できます。ラベルは一意である必要がないため、このラベルを使用してキーをグループ化および分類できます。

必須: はい

-id

キーペアのユーザー定義識別子 (ID) を指定します。クラスター内で一意の文字列を入力します。デフォルトは空の文字列です。指定した ID は、ペアの両方のキーに適用されます。

デフォルト: ID 値なし。

必須: いいえ

-min_srv

`-timeout` パラメーターの値が期限切れになる前に、キーが同期される HSM の最小数を指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、`min_srv` の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-m_value

ペアの秘密キーを使用する暗号化オペレーションを承認する必要があるユーザーの数を指定します。`0` から `8` までの値を入力します。

このパラメータにより、プライベートキーのクォーラム認証要件が確立されます。デフォルト値、`0` で、キーのクォーラム認証機能を無効にします。クォーラム認証が有効になっている場合、指定された数のユーザーがトークンに署名して、プライベートキーを使用する暗号化オペレーション、およびプライベートキーを共有または共有解除するオペレーションを承認する必要があります。

キーの `m_value` を見つけるには、getKeyInfo を使用します。

このパラメーターは、コマンドの `-u` パラメーターが `m_value` 要件を満たすのに十分なユーザーとキーペアを共有している場合にのみ有効です。

デフォルト: 0

必須: いいえ

-nex

プライベートキーを抽出できなくなります。生成されたプライベートキーを HSM からエクスポートする ことはできません。公開キーは常に抽出可能です。

デフォルト: キーペアの公開キーとプライベートキーの両方が抽出可能です。

必須: いいえ

-sess

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、setAttribute を使用します。

デフォルト: キーは永続的です。

必須: いいえ

-timeout

キーが `min_srv` パラメータで指定された HSM の数に同期されるのをコマンドが待機する時間 (秒単位) を指定します。

このパラメータは、`min_srv` パラメータがコマンドでも使用されている場合にのみ有効です。

デフォルト: タイムアウトなし。このコマンドは無期限に待機し、キーが最小数のサーバーと同期されている場合にのみ戻ります。

必須: いいえ

-u

ペアのプライベートキーを指定されたユーザーと共有します。このパラメータは、他の HSM Crypto User (CU) に暗号化オペレーションでプライベートキーを使用する許可を与えます。公開キーは、共有なしですべてのユーザーが使用可能です。

(`-u 5,6`などの) HSM ユーザー ID のカンマ区切りリストを入力します。現在のユーザーの HSM ユーザー ID を含めないでください。HSM で CU の HSM ユーザー ID を検索するには、listUsers

を使用します。既存のキーを共有および共有解除するには、cloudhsm_mgmt_util で shareKey を使用します。

デフォルト: 現在のユーザーのみがプライベートキーを使用できます。

必須: いいえ

-attest

クラスターを実行するファームウェアが改ざんされていないことを確認する整合性チェックを実行します。

デフォルト: 認証チェックなし。

必須: いいえ

関連トピック

- genSymKey
- genDSAKeyPair
- genECCKeyPair

KMU AWS CloudHSM を使用して対称キーを生成する

AWS CloudHSM key_mgmt_util ツールの genSymKey コマンドを使用して、ハードウェアセキュリティモジュール (HSM) に対称キーを生成します。キーのタイプとサイズを指定し、ID とラベルを割り当て、他の HSM ユーザーとキーを共有することができます。また、抽出不可のキーや、セッションが終了すると同時に失効するキーを作成することもできます。コマンドが成功すると、HSM がキーに割り当てるキーハンドルが返されます。このキーハンドルでキーを識別して他のコマンドで使用できます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
genSymKey -h

genSymKey -t <key-type>
          -s <key-size>
```

```
        -l <label>
        [-id <key-ID>]
        [-min_srv <minimum-number-of-servers>]
        [-m_value <0..8>]
        [-nex]
        [-sess]
        [-timeout <number-of-seconds> ]
        [-u <user-ids>]
        [-attest]
```

例

以下の例では、genSymKey を使用して HSM に対称キーを作成する方法を示しています。

> **Tip**
>
> これらの例で作成したキーをHMACオペレーションに使用するには、OBJ_ATTR_SIGN キーを生成した後に OBJ_ATTR_VERIFY と TRUE を設定する必要があります。これらの値を設定するには、CloudHSM 管理ユーティリティ (CMU) で setAttribute を使用します。詳細については、setAttribute を参照してください。

Example: AES キーの生成

このコマンドは、aes256 というラベルを持つ 256 ビット AES キーを作成します。出力は、新しいキーのキーハンドルが 6 であることを示します。

```
Command: genSymKey -t 31 -s 32 -l aes256

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 6

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: セッションキーの作成

次のコマンドは、現在のセッションでのみ有効な、抽出不可の 192 ビット AES キーを作成します。次のようなキーを作成して、エクスポートするキーをラップ (および直後にラップ解除) することができます。

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example: 迅速に戻る

次のコマンドでは、IT_test_key をラベルとする 512 バイトの汎用キーを作成します。このコマンドは、キーがクラスターのすべての HSM に同期されるまで待機しません。代わりに、いずれかの HSM でキーが作成された時点 (-min_srv 1) または 1 秒 (-timeout 1) のいずれか短い方で戻ります。タイムアウトが経過する前に、指定した最小数の HSM にキーが同期されない場合、キーは生成されません。次の例の for ループのように、多数のキーを作成するスクリプトでこのようなコマンドを使用できます。

```
Command: genSymKey -t 16 -s 512 -l IT_test_key -min_srv 1 -timeout 1

$  for i in {1..30};
     do /opt/cloudhsm/bin/key_mgmt_util singlecmd loginHSM -u CU -s example_user -p
 example_pwd genSymKey -l aes -t 31 -s 32 -min_srv 1 -timeout 1;
 done;
```

Example: クォーラム認証汎用キーの作成

次のコマンドでは、generic-mV2 をラベルとする 2048 ビットの汎用シークレットキーを作成します。このコマンドでは、-u パラメータを使用して別の CU、ユーザー 6、とキーを共有します。-m_value パラメータを使用して、キーを使用するすべての暗号オペレーションで 2 つ以上のクォーラムの承認を要求します。また、このコマンドでは、-attest パラメータを使用して、キーが生成されたファームウェアの整合性を検証します。

出力は、コマンドがキーハンドル 9 でキーを生成し、クラスターファームウェアの認証チェックが成功したことを示しています。

```
                Command:  genSymKey -t 16 -s 2048 -l generic-mV2 -m_value 2 -u 6 -
attest

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 9

        Attestation Check : [PASS]

        Cluster Error Status
```

```
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: キーの作成と検証

このコマンドは、ラベルが 3DES_shared で ID が IT-02 の Triple DES キーを作成します。現在のユーザーと、ユーザー 4 およびユーザー 5 がキーを使用できます。クラスター内で ID が一意でない場合、または現在のユーザーがユーザー 4 またはユーザー 5 の場合、コマンドは失敗します。

出力は、新規キーのキーハンドルが 7 であることを示しています。

```
Command: genSymKey -t 21 -s 24 -l 3DES_shared -id IT-02 -u 4,5

       Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 7

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

新しい 3DES キーが現在のユーザーによって所有され、ユーザー 4 とユーザー 5 と共有されていることを確認するには、getKeyInfo を使用します。このコマンドは、新しいキーに割り当てられたハンドル (Key Handle: 7) を使用します。

出力は、キーの所有者がユーザー 3 で、キーをユーザー 4 とユーザー 5 が共有していることを示しています。

```
Command:  getKeyInfo -k 7

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3

        also, shared to following 2 user(s):

                4, 5
```

キーの他のプロパティを確認するには、getAttribute を使用します。最初のコマンドでは、getAttribute を使用して、キーハンドル 7 (-o 7) のすべての属性 (-a 512) を取得します。

それを attr_7 ファイルに書き込みます。2 番目のコマンドは、cat を使用して、attr_7 ファイルの内容を取得します。

このコマンドは、キー 7 が 192 ビット (OBJ_ATTR_VALUE_LEN 0x00000018 または 24 バイト) 3DES (OBJ_ATTR_KEY_TYPE 0x15) 対称キー (OBJ_ATTR_CLASS 0x04) で、ラベルが 3DES_shared (OBJ_ATTR_LABEL 3DES_shared)、ID が IT_02 (OBJ_ATTR_ID IT-02) であることを示しています。このキーは永続的 (OBJ_ATTR_TOKEN 0x01) および抽出可能 (OBJ_ATTR_EXTRACTABLE 0x01) で、暗号化、復号、およびラッピングに使用できます。

> **Tip**
>
> タイプ、長さ、ラベル、ID など、作成したキーの属性を検索するには、getAttribute を使用します。特定のユーザーのキーを見つけるには、getKeyInfo を使用します。属性値に基づいてキーを検索するには、findKey を使用します。

キー属性の解釈については、 AWS CloudHSM KMU のキー属性リファレンス を参照してください。

```
Command:  getAttribute -o 7 -a 512 -out attr_7

got all attributes of size 444 attr cnt 17
Attributes dumped into attr_7 file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS


$  cat attr_7

OBJ_ATTR_CLASS
0x04
OBJ_ATTR_KEY_TYPE
0x15
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
```

```
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
3DES_shared
OBJ_ATTR_ID
IT-02
OBJ_ATTR_VALUE_LEN
0x00000018
OBJ_ATTR_KCV
0x59a46e
```

> **Tip**
>
> これらの例で作成したキーをHMACオペレーションに使用するには、OBJ_ATTR_SIGN キーを生成した後に OBJ_ATTR_VERIFY と TRUE を設定する必要があります。これらの値を設定するには、CMUで setAttribute を使用します。詳細については、setAttribute を参照してください。

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-t

対称キーのタイプを指定します。キーのタイプを表す定数を入力します。たとえば、AES キーを作成するには「`-t 31`」と入力します。

有効な値:

- 16: GENERIC_SECRET。汎用シークレットキー は、AES キーの要件など、特定のスタンダードに準拠していないバイト配列です。
- 18: RC4。RC4 キーは FIPS モードの HSM では無効です
- 21: Triple DES (3DES)。NIST のガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。
- 31: AES

必須: はい

-s

キーのサイズをバイト単位で指定します。たとえば、192 ビットのキーを作成するには「24」と入力します。

各キータイプに有効な値:

- AES: 16 (128 ビット)、24 (192 ビット)、32 (256 ビット)
- 3DES: 24 (192 ビット)
- 汎用シークレット: <3584 (28672 ビット)

必須: はい

-l

キーのユーザー定義ラベルを指定します。文字列を入力します。

キーを識別するのに役立つ任意のフレーズを使用できます。ラベルは一意である必要がないため、このラベルを使用してキーをグループ化および分類できます。

必須: はい

-attest

クラスターを実行するファームウェアが改ざんされていないことを確認する整合性チェックを実行します。

デフォルト: 認証チェックなし。

必須: いいえ

-id

キーのユーザー定義識別子を指定します。クラスター内で一意の文字列を入力します。デフォルトは空の文字列です。

デフォルト: ID 値なし。

必須: いいえ

-min_srv

`-timeout` パラメーターの値が期限切れになる前に、キーが同期される HSM の最小数を指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、`min_srv` の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-m_value

キーを使用する暗号化オペレーションを承認する必要があるユーザーの数を指定します。`0` から `8` までの値を入力します。

このパラメータは、キーのクォーラム認証要件を確立します。デフォルト値、`0` で、キーのクォーラム認証機能を無効にします。クォーラム認証が有効になっている場合、指定された数のユーザーは、キーを使用する暗号化オペレーション、およびキーを共有または共有解除するオペレーションを承認するためにトークンに署名する必要があります。

キーの `m_value` を見つけるには、getKeyInfo を使用します。

このパラメータが有効なのは、コマンドの `-u` パラメータが `m_value` の要件を満たすために十分な数のユーザーとキーを共有するときのみです。

デフォルト: 0

必須: いいえ

-nex

キーを抽出できなくなります。生成されたキーは HSM からエクスポートできません。

デフォルト: キーは抽出可能です。

必須: いいえ

-sess

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、setAttribute を使用します。

デフォルト: キーは永続的です。

必須: いいえ

-timeout

キーが `min_srv` パラメータで指定された HSM の数に同期されるのをコマンドが待機する時間 (秒単位) を指定します。

このパラメータは、`min_srv` パラメータがコマンドでも使用されている場合にのみ有効です。

デフォルト: タイムアウトなし。このコマンドは無期限に待機し、キーが最小数のサーバーと同期されている場合にのみ戻ります。

必須: いいえ

-u

指定されたユーザーとキーを共有します。このパラメータは、別の HSM Crypto User (CU) に、暗号化オペレーションでこのキーを使用するアクセス許可を付与します。

(`-u 5,6`などの) HSM ユーザー ID のカンマ区切りリストを入力します。現在のユーザーの HSM ユーザー ID を含めないでください。HSM で CU の HSM ユーザー ID を検索するには、listUsers を使用します。既存のキーを共有および共有解除するには、cloudhsm_mgmt_util で shareKey を使用します。

デフォルト : 現在のユーザーのみがキーを使用できます。

必須: いいえ

関連トピック

- exSymKey
- genRSAKeyPair
- genDSAKeyPair
- genECCKeyPair
- setAttribute

KMU を使用して AWS CloudHSM キー属性を取得する

AWS CloudHSM key_mgmt_util の getAttribute コマンドを使用して、 AWS CloudHSM キーの属性値の 1 つまたはすべてをファイルに書き込みます。AES キーのモジュラスなど、キータイプに指定した属性が存在しない場合は、getAttribute はエラーを返します。

キー属性はキーのプロパティです。キー属性には、キータイプ、クラス、ラベル、ID などの特性と、キーで実行できるアクション (暗号化、復号、ラップ、署名、検証など) を表す値が含まれています。

getAttribute は、所有しているキーと共有しているキーに対してのみ使用できます。このコマンドまたは cloudhsm_mgmt_util で getAttribute コマンドを実行できます。このコマンドは、クラスターのすべての HSM からキーの1つの属性値を取得し、それを stdout またはファイルに書き込みます。

属性とそれを表す定数のリストを取得するには、listAttributes コマンドを使用します。既存のキーの属性値を変更するには、key_mgmt_util の setAttribute および cloudhsm_mgmt_util の setAttribute を使用します。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
getAttribute -h

getAttribute -o <key handle>
```

```
            -a <attribute constant>
            -out <file>
```

例

以下の例では、getAttribute を使用して HSM でキーの属性を取得する方法を示します。

Example: キータイプを取得する

次の例では、AES、3DES、汎用キーなどのキータイプ、RSA または楕円曲線キーペアを取得します。

最初のコマンドでは listAttributes を実行し、キーの属性およびそれを表す定数を取得します。出力は、キータイプの定数が 256 であることを示しています。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

```
Command: listAttributes

Description
===========
The following are all of the possible attribute values for getAttributes.

      OBJ_ATTR_CLASS                = 0
      OBJ_ATTR_TOKEN                = 1
      OBJ_ATTR_PRIVATE              = 2
      OBJ_ATTR_LABEL                = 3
      OBJ_ATTR_KEY_TYPE             = 256
      OBJ_ATTR_ID                   = 258
      OBJ_ATTR_SENSITIVE            = 259
      OBJ_ATTR_ENCRYPT              = 260
      OBJ_ATTR_DECRYPT              = 261
      OBJ_ATTR_WRAP                 = 262
      OBJ_ATTR_UNWRAP               = 263
      OBJ_ATTR_SIGN                 = 264
      OBJ_ATTR_VERIFY               = 266
      OBJ_ATTR_LOCAL                = 355
      OBJ_ATTR_MODULUS              = 288
      OBJ_ATTR_MODULUS_BITS         = 289
      OBJ_ATTR_PUBLIC_EXPONENT      = 290
      OBJ_ATTR_VALUE_LEN            = 353
      OBJ_ATTR_EXTRACTABLE          = 354
      OBJ_ATTR_KCV                  = 371
```

2 番目のコマンドでは getAttribute を実行します。これは、キーハンドル 524296 のキータイプ (属性 256) をリクエストし、attribute.txt ファイルに書き込みます。

```
Command: getAttribute -o 524296 -a 256 -out attribute.txt
Attributes dumped into attribute.txt file
```

最後のコマンドでは、キーファイルの内容を取得します。出力は、キータイプが 0x15 または 21 の Triple DES (3 DES) キーであることを示しています。クラスとタイプの値の定義については、「キー属性リファレンス」を参照してください。

```
$  cat attribute.txt
OBJ_ATTR_KEY_TYPE
0x00000015
```

Example: キーのすべての属性を取得する

次のコマンドでは、キーハンドル 6 でキーのすべての属性を取得し、attr_6 ファイルに書き込みます。すべての属性を表す属性値として 512 を使用します。

```
Command: getAttribute -o 6 -a 512 -out attr_6

got all attributes of size 444 attr cnt 17
Attributes dumped into attribute.txt file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS>
```

次のコマンドでは、すべての属性の値とサンプルの属性ファイルのコンテンツを示します。値の中で、キーは 256 ビット AES キーで ID は test_01、ラベルは aes256 であることを示しています。キーは抽出可能で永続的であり、セッション専用キーではありません。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

```
$  cat attribute.txt

OBJ_ATTR_CLASS
0x04
OBJ_ATTR_KEY_TYPE
0x15
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
```

```
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
0x01
OBJ_ATTR_UNWRAP
0x01
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
aes256
OBJ_ATTR_ID
test_01
OBJ_ATTR_VALUE_LEN
0x00000020
OBJ_ATTR_KCV
0x1a4b31
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-オ

ターゲットキーのキーハンドルを指定します。各コマンドに指定できるキーは 1 つのみです。キーのキーハンドルを取得するには、findKey を使用します。

また、指定するキーは所有しているか、共有している必要があります。キーのユーザーを確認するには、getKeyInfo を使用します。

必須: はい

-a

属性を識別します。属性を表す定数を入力するか、すべての属性を表す 512 を入力します。たとえば、キーの種類を取得するには「256」と入力します。これは OBJ_ATTR_KEY_TYPE 属性を表す定数です。

属性とその定数のリスト化するために、listAttributes を使用します。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

必須: はい

-out

指定したファイルに出力を書き込みます。ファイルパスを入力します。出力を stdout に書き込むことはできません。

指定したファイルが既に存在する場合、getAttribute は警告なしにそのファイルを上書きします。

必須: はい

関連トピック

- cloudhsm_mgmt_util の中の setAttribute です。
- listAttributes
- setAttribute
- findKey
- キー属性リファレンス

KMU を使用して AWS CloudHSM キーをフェイク PEM 形式にエクスポートする

AWS CloudHSM key_mgmt_util の getCaviumPrivKey コマンドを使用して、ハードウェアセキュリティモジュール (HSM) からプライベートキーをフェイク PEM 形式でエクスポートします。フェイク PEM ファイルは、実際のプライベートキーマテリアルを含むわけではなく、HSM のプライベートキーを参照し、ウェブサーバーから AWS CloudHSMへの SSL/TLS オフロードを確立するために使用できます。詳細については、「Tomcat を使用した Linux での SSL/TLS オフロード」または「NGINX または Apache を使用した Linux での SSL/TLS オフロード」を参照してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

構文

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle>
                 -out <fake-PEM-file>
```

例

この例では、getCaviumPrivKey を使ってプライベートキーをフェイク PEM 形式でエクスポートする方法を示します。

Example: フェイク PEM ファイルをエクスポートする

このコマンドは、ハンドルが 15 のプライベートキーのフェイク PEM バージョンを作成してエクスポートし、cavKey.pem という名前のファイルを保存します。exportPrivateKey は、コマンドが成功すると成功メッセージを返します。

```
Command: getCaviumPrivKey -k 15 -out cavKey.pem

Private Key Handle is written to cavKey.pem in fake PEM format

        getCaviumPrivKey returned: 0x00 : HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-k

フェイク PEM 形式でエクスポートするプライベートキーのキーハンドルを指定します。

必須: はい

-out

フェイク PEM キーの書き込み先とするファイルの名前を指定します。

必須: はい

関連トピック

- importPrivateKey
- Tomcat を使用した Linux での SSL/TLS オフロード
- NGINX または Apache を使用した Linux での SSL/TLS オフロード

KMU を使用して HSM AWS CloudHSM パーティション証明書を取得する

AWS CloudHSM key_mgmt_util の getCert コマンドを使用して、ハードウェアセキュリティモジュールの (HSM) パーティション証明書を取得し、ファイルに保存します。コマンドを実行する際、取得する証明書のタイプを指定します。そのためには、以下の「パラメータ」セクションで説明されているように、いずれかの整数を使用します。これらの各証明書のロールについては、「HSM のアイデンティティの確認」を参照してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
getCert -h

getCert -f <file-name>
        -t <certificate-type>
```

例

この例では、getCert を使用して、クラスターのお客様のルート証明書を取得し、ファイルに保存する方法を示します。

Example: お客様のルート証明書を取得する

このコマンドは、お客様のルート証明書 (整数 4 で表現) をエクスポートし、userRoot.crt というファイルに保存します。getCert は、コマンドが成功すると成功メッセージを返します。

```
Command: getCert -f userRoot.crt -s 4

Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-f

取得された証明書の保存先とするファイルの名前を指定します。

必須: はい

-s

取得するパーティション証明書のタイプを指定する整数。整数とその証明書タイプは次のとおりです。

- 1 - 製造元のルート証明書。
- 2 - 製造元のハードウェア証明書。
- 4 - お客様のルート証明書。
- 8 - (お客様のルート証明書で署名されている)クラスターの証明書。
- 16 - (製造元のルート証明書に連鎖されている)クラスターの証明書。

必須: はい

関連トピック

- HSM のアイデンティティの確認

KMU を使用して AWS CloudHSM キーのユーザーを取得する

AWS CloudHSM key_mgmt_util の getKeyInfo コマンドを使用して、キーを共有している所有者や暗号化ユーザー (CU) など、キーを使用できるユーザーのハードウェアセキュリティモジュール (HSM) ユーザー IDs を返します。キーに対するクォーラム認証が有効になっている場合、getKeyInfo はキーを使用する暗号化オペレーションを承認する必要があるユーザーの数も返します。getKeyInfo は、所有および共有しているキーに対してのみ実行できます。

パブリックキーに対して getKeyInfo を実行すると、HSM のすべてのユーザーがパブリックキーを使用できる場合でも、getKeyInfo はキー所有者のみを返します。HSM のユーザーの HSM ユーザー ID を確認するには、listUsers を使用します。特定のユーザーのキーを確認するには、findKey -u を使用します。

ユーザーは、自分で作成したキーを所有します。自分で作成したキーは、他のユーザーと共有できます。既存のキーを共有または共有解除するには、cloudhsm_mgmt_util の shareKey を使用します。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする必要があります。

Syntax

```
getKeyInfo -h

getKeyInfo -k <key-handle>
```

例

以下の例では、getKeyInfo を使用してキーのユーザーに関する情報を取得する方法を示します。

Example: 対称キーのユーザーを取得する

次のコマンドでは、キーハンドルが 9 の AES (対称) キーを使用できるユーザーを取得します。出力は、キーの所有者がユーザー 3 であり、キーをユーザー 4 と共有していることを示しています。

```
Command:  getKeyInfo -k 9

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 1 user(s):

                4
```

Example: 非対称キーペアのユーザーを取得する

以下のコマンドでは、getKeyInfo を使用して RSA (非対称) キーペアのキーを使用できるユーザーを取得します。パブリックキーのキーハンドルは 21 です。プライベートキーのキーハンドルは 20 です。

プライベートキー (getKeyInfo) に対して 20 を実行すると、キー所有者 (3) およびキーを共有している Crypto User (CU) 4 と 5 が返されます。

```
Command:  getKeyInfo -k 20

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 2 user(s):

                4
                5
```

getKeyInfo をパブリックキー (21) に対して実行すると、キー所有者 (3) のみが返されます。

```
Command:  getKeyInfo -k 21

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3
```

ユーザー 4 がパブリックキー (および HSM のすべてのパブリックキー) を使用できることを確認するには、-ufindKey の パラメータを使用します。

出力は、ユーザー 4 がキーペアのパブリックキー (21) とプライベートキー (20) の両方を使用できることを示しています。ユーザー 4 は、他のすべてのパブリックキーと、自分で作成したプライベートキーまたは共有しているプライベートキーを使用することもできます。

```
Command:  findKey -u 4
Total number of keys present 8

 number of keys matched from start index 0::7
11, 12, 262159, 262161, 262162, 19, 20, 21

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example: キーのクォーラム認証値 (m_value) を取得する

この例では、キーの m_value、つまりキーを使用する暗号化オペレーションを承認する必要があるクォーラムのユーザー数を取得する方法を示します。

キーに対してクォーラム認証を有効にすると、ユーザーのクォーラムは、そのキーを使用する暗号化オペレーションを承認する必要があります。クォーラム認証を有効にしてクォーラムサイズを設定するには、キーの作成時に -m_value パラメータを使用します。

次のコマンドでは、genRSAKeyPair を使用して、ユーザー 4 と共有される RSA キーペアを作成します。また、m_value パラメータを使用してペアのプライベートキーでクォーラム認証を有効にし、クォーラムサイズを 2 ユーザーに設定します。ユーザー数は必要な承認を提供できるだけの大きさが必要です。

出力は、このコマンドでパブリックキー 27 とプライベートキー 28 が作成されたことを示しています。

```
Command:  genRSAKeyPair -m 2048 -e 195193 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2

       Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

       Cfm3GenerateKeyPair:    public key handle: 27    private key handle: 28

       Cluster Error Status
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
       Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

次のコマンドでは、getKeyInfo を使用して、プライベートキーのユーザーに関する情報を取得します。出力は、キーの所有者がユーザー 3 であり、キーがユーザー 4 と共有されていることを示しています。また、2 ユーザーのクォーラムが、このキーを使用するすべての暗号化オペレーションを承認する必要があることも示しています。

```
Command:  getKeyInfo -k 28

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 1 user(s):

                4
```

```
        2 Users need to approve to use/manage this key
```

パラメータ

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-k

HSM で 1 つのキーのキーハンドルを指定します。所有または共有するキーのキーハンドルを入力します。このパラメータは必須です。

キーハンドルを見つけるには、findKey コマンドを使用します。

必須: はい

関連トピック

- cloudhsm_mgmt_util の getKeyInfo
- listUsers
- findKey
- cloudhsm_mgmt_util の中の findAllKeys です。

KMU AWS CloudHSM のヘルプ情報を表示する

AWS CloudHSM key_mgmt_util の help コマンドを使用して、使用可能なすべての key_mgmt_util コマンドに関する情報を表示します。

help を実行する前に、key_mgmt_util を起動する必要があります。

Syntax

```
help
```

例

次の例は、help コマンドの出力を示します。

Example

```
Command:  help

Help Commands Available:

Syntax: <command> -h


   Command               Description
   =======               ===========

   exit                   Exits this application
   help                   Displays this information

        Configuration and Admin Commands
   getHSMInfo             Gets the HSM Information
   getPartitionInfo       Gets the Partition Information
   listUsers              Lists all users of a partition
   loginStatus            Gets the Login Information
   loginHSM               Login to the HSM
   logoutHSM              Logout from the HSM

        M of N commands
   getToken               Initiate an MxN service and get Token
   delToken               delete Token(s)
   approveToken           Approves an MxN service
   listTokens             List all Tokens in the current partition

        Key Generation Commands

        Asymmetric Keys:
   genRSAKeyPair          Generates an RSA Key Pair
   genDSAKeyPair          Generates a DSA Key Pair
   genECCKeyPair          Generates an ECC Key Pair

        Symmetric Keys:
   genPBEKey              Generates a PBE DES3 key
   genSymKey              Generates a Symmetric keys

        Key Import/Export Commands
   createPublicKey        Creates an RSA public key
   importPubKey           Imports RSA/DSA/EC Public key
   exportPubKey           Exports RSA/DSA/EC Public key
```

```
    importPrivateKey       Imports RSA/DSA/EC private key
    exportPrivateKey       Exports RSA/DSA/EC private key
    imSymKey               Imports a Symmetric key
    exSymKey               Exports a Symmetric key
    wrapKey                Wraps a key from from HSM using the specified handle
    unWrapKey              UnWraps a key into HSM using the specified handle

         Key Management Commands
    deleteKey              Delete Key
    setAttribute           Sets an attribute of an object
    getKeyInfo             Get Key Info about shared users/sessions
    findKey                Find Key
    findSingleKey          Find single Key
    getAttribute           Reads an attribute from an object

         Certificate Setup Commands
    getCert                Gets Partition Certificates stored on HSM

         Key Transfer Commands
    insertMaskedObject     Inserts a masked object
    extractMaskedObject    Extracts a masked object

         Management Crypto Commands
    sign                   Generates a signature
    verify                 Verifies a signature
    aesWrapUnwrap          Does NIST AES Wrap/Unwrap

         Helper Commands
    Error2String           Converts Error codes to Strings
                           save key handle in fake PEM format
    getCaviumPrivKey       Saves an RSA private key handle
                           in fake PEM format
    IsValidKeyHandlefile   Checks if private key file has
                           an HSM key handle or a real key
    listAttributes         List all attributes for getAttributes
    listECCCurveIds        List HSM supported ECC CurveIds
```

パラメータ

このコマンドにはパラメータがありません。

関連トピック

- loginHSM および logoutHSM

KMU AWS CloudHSM を使用してプライベートキーをインポートする

AWS CloudHSM key_mgmt_util の importPrivateKey コマンドを使用して、非対称プライベートキーをファイルからハードウェアセキュリティモジュール (HSM) にインポートします。HSM では、クリアテキストのキーを直接インポートすることはできません。このコマンドは、指定した AES ラップキーを使用してプライベートキーを暗号化し、HSM 内のキーをラップ解除します。 AWS CloudHSM キーを証明書に関連付ける場合は、このトピックを参照してください。

> **Note**
>
> パスワードで保護された PEM キーは、対称キーまたはプライベートキーを使用してインポートすることはできません。

OBJ_ATTR_UNWRAP と OBJ_ATTR_ENCRYPT 属性値 1 を含む AES ラップキーを指定する必要があります。キーの属性を確認するには、getAttribute コマンドを使用します。

> **Note**
>
> このコマンドでは、インポートしたキーをエクスポート不可として指定するオプションは使用できません。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
importPrivateKey -h

importPrivateKey -l <label>
                 -f <key-file>
                 -w <wrapping-key-handle>
                 [-sess]
                 [-id <key-id>]
                 [-m_value <0...8>]
                 [min_srv <minimum-number-of-servers>]
                 [-timeout <number-of-seconds>]
                 [-u <user-ids>]
                 [-wk <wrapping-key-file>]
```

```
                [-attest]
```

例

この例では、importPrivateKey を使ってプライベートキーを HSM にインポートする方法を示します。

Example: プライベートキーをインポートする

このコマンドは、rsa2048.key というラベルとハンドルが 524299 のラップキーを使って、rsa2048-imported という名前のファイルからプライベートキーをインポートします。importPrivateKey コマンドは、成功すると、インポートされたキーのキーハンドルと成功メッセージを返します。

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299

BER encoded key length is 1216

Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Private Key Unwrapped.  Key Handle: 524301

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-l

ユーザー定義のプライベートキーラベルを指定します。

必須: はい

-f

インポートするキーのファイル名を指定します。

必須: はい

-w

ラップキーのキーハンドルを指定します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

キーをラップキーとして使用できるかどうかを確認するには、getAttribute を使用して OBJ_ATTR_WRAP 属性 (262) の値を取得します。ラップキーを作成するには、genSymKey を使用して AES キー (タイプ 31) を作成します。

-wk パラメータを使用して外部のラップ解除キーを指定した場合、インポート時の (ラップ解除ではなく) ラップには -w ラップキーが使われます。

必須: はい

-sess

インポートされたキーをセッションキーに指定します。

デフォルト: インポートされたキーは、クラスター内で永続 (トークン) キーとして保持されます。

必須: いいえ

-id

インポートするキーの ID を指定します。

デフォルト: ID 値なし。

必須: いいえ

-m_value

インポートされたキーを使用した暗号化オペレーションを、何人のユーザーが承認しなければならないかを指定します。**0** ~ **8** の値を入力します。

このパラメータが有効なのは、コマンドの -u パラメータが m_value の要件を満たすために十分な数のユーザーとキーを共有するときのみです。

デフォルト: 0

必須: いいえ

-min_srv

-timeout パラメータの値が期限切れになる前に、インポートされたキーが最小いくつの HSM で同期されるかを指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、min_srv の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-timeout

min-serv パラメータが含まれている場合に、すべての HSM でキーが同期されるまで待機する秒数を指定します。数値が指定されていない場合、ポーリングが永遠に続きます。

デフォルト: 無制限

必須: いいえ

-u

インポートされたプライベートキーを共有するユーザーのリストを指定します。このパラメータは、他の HSM Crypto User (CU) に対し、インポートされたキーを暗号化オペレーションに使用するアクセス許可を付与します。

HSM ユーザー ID のカンマ区切りリスト (例: --u 5,6) を入力します。現在のユーザーの HSM ユーザー ID を含めないでください。HSM で CU の HSM ユーザー ID を検索するには、listUsers を使用します。

デフォルト: 現在のユーザーのみがインポートされたキーを使用できます。

必須: いいえ

-wk

インポートするキーのラップ解除に使用するキーを指定します。プレーンテキストの AES キーが含まれているファイルのパスと名前を入力します。

このパラメータを含めた場合、importPrivateKey は、インポートするキーのラップに -wk ファイルのキーを使用します。また、ラップ解除には -w パラメータで指定されたキーを使用します。

デフォルト: -w パラメータで指定されたラップキーを使用して、ラップとラップ解除の両方を行う。

必須: いいえ

-attest

ファームウェアレスポンスの証明チェックを実行し、クラスターを実行するファームウェアが侵害されていないことを確認します。

必須: いいえ

関連トピック

- wrapKey
- unWrapKey
- genSymKey
- exportPrivateKey

KMU AWS CloudHSM を使用してパブリックキーをインポートする

AWS CloudHSM key_mgmt_util の importPubKey コマンドを使用して、PEM 形式のパブリックキーをハードウェアセキュリティモジュール (HSM) にインポートします。このコマンドを使用すると、HSM の外部で生成されたパブリックキーをインポートできます。また、このコマンドを使用して、exportPubKey コマンドでエクスポートされたキーなど、HSM からエクスポートされたキーをインポートすることもできます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
importPubKey -h

importPubKey -l <label>
             -f <key-file>
             [-sess]
```

```
            [-id <key-id>]
            [min_srv <minimum-number-of-servers>]
            [-timeout <number-of-seconds>]
```

例

この例では、importPubKey を使ってパブリックキーを HSM にインポートする方法を示します。

Example: パブリックキーをインポートする

このコマンドは、importedPublicKey というラベルを使って public.pem という名前のファイルからパブリックキーをインポートします。importPubKey コマンドは、成功すると、インポートされたキーのキーハンドルと成功メッセージを返します。

```
Command: importPubKey -l importedPublicKey -f public.pem

Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS

Public Key Handle: 262230

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-l

ユーザー定義のパブリックキーラベルを指定します。

必須: はい

-f

インポートするキーのファイル名を指定します。

必須: はい

-sess

インポートされたキーをセッションキーに指定します。

デフォルト: インポートされたキーは、クラスター内で永続 (トークン) キーとして保持されます。

必須: いいえ

-id

インポートするキーの ID を指定します。

デフォルト: ID 値なし。

必須: いいえ

-min_srv

`-timeout` パラメータの値が期限切れになる前に、インポートされたキーが最小いくつの HSM に同期されるかを指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、`min_srv` の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-timeout

`min-serv` パラメータが含まれている場合に、すべての HSM でキーが同期されるまで待機する秒数を指定します。数値が指定されていない場合、ポーリングが永遠に続きます。

デフォルト: 無制限

必須: いいえ

関連トピック

- exportPubKey

- キーの生成

AWS CloudHSM KMU を使用してプレーンテキストの対称キーをインポートする

AWS CloudHSM key_mgmt_util ツールの imSymKey コマンドを使用して、対称キーのプレーンテキストコピーをファイルからハードウェアセキュリティモジュール (HSM) にインポートします。これを使用して、HSM 外で任意の方法で生成したキーや、HSM からエクスポートしたキー (exSymKey でファイルに書き込んだキーなど) をインポートできます。

インポートプロセス中に、imSymKey は選択した AES キー (ラッピングキー) を使用して、インポートするキーをラップ (暗号化) してからラップ解除 (復号化) します。ただし、imSymKey を使用できるのは、プレーンテキストのキーが含まれているファイルに対してのみです。暗号化されたキーのエクスポートとインポートには、wrapKey コマンドと unWrapKey コマンドを使用します。

また、imSymKey コマンドは対称キーのみをインポートします。パブリックキーをインポートするには、importPubKey を使用します。プライベートキーをインポートするには、 importPrivateKey または wrapKey を使用します。

> **Note**
>
> パスワードで保護された PEM キーは、対称キーまたはプライベートキーを使用してインポートすることはできません。

インポートしたキーは、HSM で生成したキーとほぼ同じように動作します。ただし、OBJ_ATTR_LOCAL 属性の値は 0 であり、ローカルに生成されたものでないことを示しています。次のコマンドを使用して、インポートした対称キーを共有します。shareKeycloudhsm_mgmt_util で コマンドを使用して、インポート後にキーを共有します。

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

キーのインポート後に、必ずキーファイルをマークまたは削除してください。このコマンドでは、同じキーマテリアルを複数回インポートすることが禁止されません。その結果、キーハンドルが異なる複数のキーが同じキーマテリアルを持つ場合があり、キーマテリアルの使用の追跡が困難になります。また、暗号化の制限に制約されます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
imSymKey -h

imSymKey -f <key-file>
         -w <wrapping-key-handle>
         -t <key-type>
         -l <label>
         [-id <key-ID>]
         [-sess]
         [-wk <wrapping-key-file> ]
         [-attest]
         [-min_srv <minimum-number-of-servers>]
         [-timeout <number-of-seconds> ]
         [-u <user-ids>]
```

例

以下の例では、imSymKey を使用して対称キーを HSM 内にインポートする方法を示します。

Example: AES 対称キーをインポートする

次の例では、imSymKey を使用して AES 対称キーを HSM 内にインポートします。

最初のコマンドでは、OpenSSL を使用してランダムな 256 ビット AES 対称キーを生成します。生成したキーは、aes256.key ファイルに保存されます。

```
$  openssl rand -out aes256.key 32
```

2 番目のコマンドでは、imSymKey を使用し、AES キーを aes256.key ファイルから HSM 内にインポートします。HSM の AES キー (キー 20) をラップキーとして使用し、imported をラベルとして指定します。ID とは異なり、ラベルはクラスター内で一意である必要はありません。-t (タイプ) パラメータの値は、AES を表す 31 です。

出力は、ファイルのキーがラップ/ラップ解除され、次に HSM 内にインポートされて、キーハンドル 262180 が割り当てられたことを示しています。

```
Command:  imSymKey -f aes256.key -w 20 -t 31 -l imported

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 262180

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

次のコマンドでは、getAttribute を使用して、新しくインポートしたキーの OBJ_ATTR_LOCAL 属性 (属性 355) を取得し、それを attr_262180 ファイルに書き込みます。

```
Command:  getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

属性ファイルを調べると、OBJ_ATTR_LOCAL 属性の値は 0 であり、キーマテリアルが HSM で生成されたものでないことがわかります。

```
$  cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example: クラスター間で対称キーを移動する

次の例では、exSymKey と imSymKey を使用し、クラスター間でプレーンテキストの AES キーを移動する方法を示します。次のようなプロセスを使用して HSM の両クラスターで有効な AES ラッピングを作成できます。共有ラップキーの準備が整ったら、wrapKey と unWrapKey を使用してクラスター間で暗号化されたキーを移動できます。

このオペレーションを実行する CU ユーザーには、両クラスターで HSM にログインするアクセス許可が必要です。

最初のコマンドでは、exSymKey を使用し、キー 14 (32 ビット AES キー) をクラスター 1 から aes.key ファイル内にエクスポートします。ラップキーとして、クラスター 1 の HSM でキー 6 (AES キー) を使用します。

```
Command: exSymKey -k 14 -w 6 -out aes.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "aes.key"
```

次に、ユーザーはクラスター 2 の key_mgmt_util にログインし、imSymKey コマンドを実行して aes.key ファイルのキーをクラスター 2 の HSM 内にインポートします。このコマンドでは、ラップキーとして、クラスター 2 の HSM で キー 252152 (AES キー) を使用します。

exSymKey と imSymKey で使用するラップキーは、ターゲットキーをラップして即座にラップ解除するため、クラスターごとに別のものを使用できます。

出力は、キーがクラスター 2 に正常にインポートされてキーハンドル 21 が割り当てられたことを示しています。

```
Command:  imSymKey -f aes.key -w 262152 -t 31 -l xcluster

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 21

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

クラスター 1 のキー 14 とクラスター 2 のキー 21 で、キーマテリアルが同じであることを確認するには、各キーのキーチェック値 (KCV) を取得します。KCV 値が同じであれば、キーマテリアルは同じです。

次のコマンドでは、クラスター 1 の getAttribute を使用してキー 14 の KCV 属性 (属性 371) の値を attr_14_kcv ファイルに書き込みます。次に、cat コマンドを使用して、attr_14_kcv ファイルの内容を取得します。

```
Command:  getAttribute -o 14 -a 371 -out attr_14_kcv
Attributes dumped into attr_14_kcv file

$  cat attr_14_kcv
OBJ_ATTR_KCV
0xc33cbd
```

次の同様のコマンドでは、クラスター 2 の getAttribute を使用してキー 21 の KCV 属性 (属性 371) の値を attr_21_kcv ファイルに書き込みます。次に、cat コマンドを使用して、attr_21_kcv ファイルの内容を取得します。

```
Command:  getAttribute -o 21 -a 371 -out attr_21_kcv
Attributes dumped into attr_21_kcv file

$  cat attr_21_kcv
OBJ_ATTR_KCV
0xc33cbd
```

出力は、2 つのキーの KCV 値が同じであり、キーマテリアルが同じであることを示しています。

両クラスターの HSM でキーマテリアルが同じであるため、プレーンテキストキーを公開することなく、クラスター間で暗号化されたキーを共有できます。たとえば、wrapKey コマンドでラップキー 14 を使用してクラスター 1 から暗号化されたキーをエクスポートし、次に unWrapKey でラップキー 21 を使用してクラスター 2 に暗号化されたキーをインポートできます。

Example: セッションキーをインポートする

次のコマンドでは、-sess の imSymKey パラメータを使用し、現在のセッションでのみ有効な 192 ビット Triple DES キーをインポートします。

このコマンドでは、インポートするキーが含まれているファイルを -f パラメータで指定します。また、キーのタイプを -t パラメータで指定し、ラップキーを -w パラメータで指定します。キーを分類するラベルを -l パラメータで指定し、キーのフレンドリーな一意の識別子を -id パラメータで作成します。さらに、キーをインポートするファームウェアを -attest パラメータで検証します。

出力は、キーが正常にラップ/ラップ解除され、HSM 内にインポートされて、キーハンドル 37 が割り当てられたことを示しています。また、認証チェックに合格し、ファームウェアが改ざんされていないことを示しています。

```
Command:  imSymKey -f 3des192.key -w 6 -t 21 -l temp -id test01 -sess -attest
```

```
        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 37

        Attestation Check : [PASS]

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

次に、getAttribute コマンドまたは findKey コマンドを使用し、新しくインポートされたキーの属性を検証できます。次のコマンドでは、findKey を使用して、キー 37 のタイプ、ラベル、および ID がコマンドで指定されたとおりであること、さらにセッションキーであることを検証します。出力の 5 行目が示すように、findKey は、すべての属性に一致するキーがキー 37 のみであることを示しています。

```
Command:  findKey -t 21 -l temp -id test01 -sess 1
Total number of keys present 1

 number of keys matched from start index 0::0
37

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

パラメータ

-attest

クラスターを実行するファームウェアが改ざんされていないことを確認する整合性チェックを実行します。

デフォルト: 認証チェックなし。

必須: いいえ

-f

インポートするキーが含まれているファイルを指定します。

ファイルには、指定された長さの AES キーまたは Triple DES キーのプレーンテキストコピーが含まれている必要があります。RC4 キーと DES キーは FIPS モードの HSM では無効です。

- AES: 16、24、または 32 バイト
- Triple DES (3 DES): 24 バイト

必須: はい

-h

コマンドに関するヘルプを表示します。

必須: はい

-id

キーのユーザー定義識別子を指定します。クラスター内で一意の文字列を入力します。デフォルトは空の文字列です。

デフォルト: ID 値なし。

必須: いいえ

-l

キーのユーザー定義ラベルを指定します。文字列を入力します。

キーを識別するのに役立つ任意のフレーズを使用できます。ラベルは一意である必要がないため、このラベルを使用してキーをグループ化および分類できます。

必須: はい

-min_srv

`-timeout` パラメーターの値が期限切れになる前に、キーが同期される HSM の最小数を指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、`min_srv` の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-sess

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、setAttribute を使用します。

デフォルト: キーは永続的です。

必須: いいえ

-timeout

キーが `min_srv` パラメータで指定された HSM の数に同期されるのをコマンドが待機する時間 (秒単位) を指定します。

このパラメータは、`min_srv` パラメータがコマンドでも使用されている場合にのみ有効です。

デフォルト: タイムアウトなし。このコマンドは無期限に待機し、キーが最小数のサーバーと同期されている場合にのみ戻ります。

必須: いいえ

-t

対称キーのタイプを指定します。キーのタイプを表す定数を入力します。たとえば、AES キーを作成するには「`-t 31`」と入力します。

有効な値:

- 21: Triple DES (3DES)。
- 31: AES

必須: はい

-u

指定したユーザーとインポートするキーを共有します。このパラメータは、別の HSM Crypto User (CU) に、暗号化オペレーションでこのキーを使用するアクセス許可を付与します。

1 つの ID または HSM ユーザー ID のカンマ区切りリスト (5,6 など) を入力します。現在のユーザーの HSM ユーザー ID を含めないでください。ID を確認するには、cloudhsm_mgmt_util コマンドラインツールの listUsers コマンドまたは key_mgmt_util コマンドラインツールの listUsers コマンドを使用できます。

必須: いいえ

-w

ラップキーのキーハンドルを指定します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

ラップキーは、インポートプロセスでキーの暗号化 (ラップ) と復号 (ラップ解除) に使用する HSM のキーです。ラップキーとして使用できるのは AES キーのみです。

任意の AES キー (任意のサイズ) をラップキーとして使用できます。ラップキーは、ターゲットキーをラップし、直後にラップ解除するため、セッション専用の AES キーをラップキーとして使用できます。キーをラップキーとして使用できるかどうかを確認するには、getAttribute を使用して OBJ_ATTR_WRAP 属性 (262) の値を取得します。ラップキーを作成するには、genSymKey を使用して AES キー (タイプ 31) を作成します。

-wk パラメータを使用して外部のラップキーを指定した場合、インポートしたキーは -w ラップキーでラップ解除することはできますが、ラップすることはできません。

> **Note**
>
> キー 4 は、サポートされていない内部キーです。AES キーをラップキーとして作成および管理することをお勧めします。

必須: はい

-wk

指定されたファイルの AES キーを使用して、インポートするキーをラップします。プレーンテキストの AES キーが含まれているファイルのパスと名前を入力します。

このパラメータを含めると、imSymKey は、-wk ファイルのキーを使用してインポートされたキーをラップし、-w パラメータで指定された HSM のキーを使用してラップ解除します。-w パラメータと -wk パラメータの値は同じプレーンテキストのキーに解決される必要があります。

デフォルト: HSM のラップキーを使用してラップ解除します。

必須: いいえ

関連トピック

- genSymKey
- exSymKey
- wrapKey
- unWrapKey
- exportPrivateKey
- exportPubKey

KMU AWS CloudHSM を使用してマスクされたオブジェクトを挿入する

AWS CloudHSM key_mgmt_util の insertMaskedObject コマンドを使用して、マスクされたオブジェクトをファイルから指定されたハードウェアセキュリティモジュール (HSM) に挿入します。マスクされたオブジェクトとは、クローンされたオブジェクトで、extractMaskedObject コマンドを使用して HSM から抽出されたものです。マスクされたオブジェクトは、再び元のクラスターに挿入して初めて使用可能になります。マスクされたオブジェクトは、生成元であるクラスター、またはそのクラスターのクローンにしか挿入できません。これには、リージョン間でのバックアップのコピーによって生成されたクラスターのクローンバージョンや、そのバックアップを使って新しいクラスターを作成することで生成された元のクラスターのクローンバージョンが含まれます。

マスクされたオブジェクトは、抽出不可能なキー (OBJ_ATTR_EXTRACTABLE 値が 0 であるキー) を含め、キーを効率的にオフロードおよび同期する手段です。これにより、 AWS CloudHSM 設定ファイルを更新することなく、異なるリージョンの関連するクラスター間でキーを安全に同期できます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                   [-min_srv <minimum-number-of-servers>]
                   [-timeout <number-of-seconds>]
```

例

この例では、insertMaskedObject を使ってマスクされたオブジェクトを HSM に挿入する方法を示します。

Example: マスクされたオブジェクトを挿入する

このコマンドは、maskedObj というファイルにあるマスクされたオブジェクトを HSM に挿入します。insertMaskedObject コマンドは、成功すると、マスクされたオブジェクトから複合されたキーのキーハンドルと成功メッセージを返します。

```
Command: insertMaskedObject -f maskedObj

Cfm3InsertMaskedObject returned: 0x00 : HSM Return: SUCCESS
        New Key Handle: 262433

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-f

マスクされたオブジェクトの挿入先とするファイル名を指定します。

必須: はい

-min_srv

-timeout パラメータの値が期限切れになる前に、挿入したマスクされたオブジェクトが最小いくつの HSM で同期されるかを指定します。オブジェクトが割り当てられた時間内に指定の数のサーバーに同期されなかった場合、そのオブジェクトは挿入されません。

デフォルト: 1

必須: いいえ

-timeout

min-serv パラメータが含まれている場合に、すべてのサーバーでキーが同期されるまで待機する秒数を指定します。数値が指定されていない場合、ポーリングが永遠に続きます。

デフォルト: 無制限

必須: いいえ

関連トピック

- extractMaskedObject
- syncKey
- リージョン間のバックアップのコピー
- 以前のバックアップからの AWS CloudHSM クラスターの作成

KMU AWS CloudHSM を使用してキーファイルを検証する

AWS CloudHSM key_mgmt_util の IsValidKeyHandlefile コマンドを使用して、キーファイルに実際のプライベートキーまたはフェイク RSA PEM キーが含まれているかどうかを確認します。フェイク PEM ファイルは、実際のプライベートキーマテリアルを含まず、HSM のプライベートキーを参照します。このようなファイルは、ウェブサーバーから AWS CloudHSMへの SSL/TLS オフロードを確立するために使います。詳細については、「Tomcat を使用した Linux での SSL/TLS オフロード」または「NGINX または Apache を使用した Linux での SSL/TLS オフロード」を参照してください。

> **Note**
>
> IsValidKeyHandlefile は RSA キーにのみ機能します。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
IsValidKeyHandlefile -h
```

```
IsValidKeyHandlefile -f <rsa-private-key-file>
```

例

以下の例では、IsValidKeyHandlefile を使って、あるキーファイルに含まれるのが実際のキーマテリアルなのか、フェイク PEM キーマテリアルなのかを特定する方法を示します。

Example: 実際のプライベートキーを検証する

次のコマンドは、privateKey.pem というファイルに実際のキーマテリアルが含まれていることを確認します。

```
Command: IsValidKeyHandlefile -f privateKey.pem

Input key file has real private key
```

Example: フェイク PEM キーを無効化する

次のコマンドは、caviumKey.pem というファイルにキーハンドル 15 から生成されたフェイク PEM キーが含まれていることを確認します。

```
Command: IsValidKeyHandlefile -f caviumKey.pem

Input file has invalid key handle: 15
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-h

コマンドのコマンドラインヘルプを表示します

必須: はい

-f

有効なキーマテリアルの存在を確認する RSA プライベートキーファイルを指定します。

必須: はい

関連トピック

- getCaviumPrivKey
- Tomcat を使用した Linux での SSL/TLS オフロード
- NGINX または Apache を使用した Linux での SSL/TLS オフロード

KMU を使用して AWS CloudHSM キーの属性を一覧表示する

AWS CloudHSM key_mgmt_util の listAttributes コマンドを使用して、 AWS CloudHSM キーの属性とそれを表す定数を一覧表示します。これらの定数は、getAttribute コマンドおよび setAttribute コマンドの属性を特定するのに使用します。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

このコマンドにはパラメータはありません。

```
listAttributes
```

例

このコマンドは、key_mgmt_utilで取得および変更できるキー属性と、それらを表す定数を一覧表示します。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

key_mgmt_util の getAttribute コマンドですべての属性を表すには、512 を使用します。

```
Command: listAttributes

Following are the possible attribute values for getAttributes:

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ENCRYPT                = 260
      OBJ_ATTR_DECRYPT                = 261
```

```
    OBJ_ATTR_WRAP                   = 262
    OBJ_ATTR_UNWRAP                 = 263
    OBJ_ATTR_SIGN                   = 264
    OBJ_ATTR_VERIFY                 = 266
    OBJ_ATTR_LOCAL                  = 355
    OBJ_ATTR_MODULUS                = 288
    OBJ_ATTR_MODULUS_BITS           = 289
    OBJ_ATTR_PUBLIC_EXPONENT        = 290
    OBJ_ATTR_VALUE_LEN              = 353
    OBJ_ATTR_EXTRACTABLE            = 354
    OBJ_ATTR_KCV                    = 371
```

関連トピック

- cloudhsm_mgmt_util の setAttribute
- getAttribute
- setAttribute
- キー属性リファレンス

KMU を使用するすべての AWS CloudHSM ユーザーを一覧表示する

AWS CloudHSM key_mgmt_util の listUsers コマンドを使用して、ハードウェアセキュリティモジュール (HSM) のユーザーとそのユーザータイプおよびその他の属性を取得します。

key_mgmt_util で、listUsers は、一貫性がない場合でも、クラスターのすべての HSM を表す出力を返します。各 HSM のユーザーに関する情報を取得するには、cloudhsm_mgmt_util の listUsers コマンドを使用します。

key_mgmt_util listUsers および getKeyInfo のユーザーコマンドは、Crypto User (CU) が実行する権限を持つ読み取り専用コマンドです。残りのユーザー管理コマンドは cloudhsm_mgmt_util の一部です。それらは、ユーザー管理アクセス権限を持つ Crypto Officer (CO) によって実行されます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
listUsers

listUsers -h
```

例

このコマンドは、クラスターの HSM のユーザーとその属性を一覧表示します。User ID 属性を使用して、findKey、getAttribute、および getKeyInfo などの他のコマンドでユーザーを特定できます。

```
Command:  listUsers

        Number Of Users found 4

        Index       User ID     User Type       User Name           MofnPubKey
 LoginFailureCnt         2FA
        1               1       PCO             admin                   NO
    0               NO
        2               2       AU              app_user                NO
    0               NO
        3               3       CU              alice                   YES
    0               NO
        4               4       CU              bob                     NO
    0               NO
        5               5       CU              trent                   YES
    0               NO

        Cfm3ListUsers returned: 0x00 : HSM Return: SUCCESS
```

この出力が示すユーザー属性は以下のとおりです。

- ユーザー ID: key_mgmt_util および cloudhsm_mgmt_util のコマンドでユーザーを識別します。
- User type: HSM でユーザーが実行できるオペレーションを決定します。
- User Name: ユーザー定義のわかりやすいユーザー名を表示します。
- MofnPubKey: ユーザーがクォーラム認証トークンに署名するためのキーペアを登録済みであるかどうかを示します。
- LoginFailureCnt: ユーザーがログインに失敗した回数を示します。
- 2FA: ユーザーが多要素認証を有効にしていることを示します。

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

関連トピック

- cloudhsm_mgmt_util の listUsers
- findKey
- getAttribute
- getKeyInfo

KMU を使用して HSM AWS CloudHSM にログインおよびログアウトする

AWS CloudHSM key_mgmt_util の loginHSM および logoutHSM コマンドを使用して、クラスター内のハードウェアセキュリティモジュール (HSM) にログインおよびログアウトします。HSM にログインすると、key_mgmt_util を使用して、公開キーとプライベートキーの生成、同期、ラッピングなど、さまざまなキー管理オペレーションを実行できます。

どの key_mgmt_util コマンドを実行する場合でも、事前に key_mgmt_util を起動する必要があります。key_mgmt_util を使用してキーを管理するには、Crypto User (CU) として HSM にログインする必要があります。

> **Note**
>
> ログイン試行回数が 5 回を超えると、アカウントがロックアウトされます。2018 年 2 月より前にクラスターを作成した場合、ロックアウトされるまでのログイン試行回数は 20 回です。アカウントのロックを解除するには、暗号化オフィサー (CO) が cloudhsm_mgmt_util で changePswd コマンドを使用してパスワードをリセットする必要があります。
> クラスター内に複数の HSM がある場合は、アカウントがロックアウトされるまでのログイン試行回数の上限が増える可能性があります。これは、CloudHSM クライアントがさまざまな HSM 間で負荷を分散するためです。したがって、ログイン試行は毎回同じ HSM で開始されない場合があります。この機能をテストしている場合は、アクティブな HSM が1つだけのクラスターでテストすることをお勧めします。

Syntax

```
loginHSM -h
```

```
loginHSM -u <user type>
         { -p | -hpswd } <password>
         -s <username>
```

例

この例では、loginHSM および logoutHSM コマンドを使ってクラスターの HSM でログインおよびログアウトする方法を示します。

Example: HSM にログインする

このコマンドは、CU というユーザー名と example_user というパスワードを使い、Crypto User (aws) として HSM にログインします。出力には、クラスターのすべてのHSMにログインしたことが示されます。

```
Command:  loginHSM -u CU -s example_user -p aws

Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: 隠しパスワードでログインします。

このコマンドは上記の例と同じですが、今回はシステムがパスワードを隠すように指定することを除きます。

```
Command:  loginHSM -u CU -s example_user -hpswd
```

システムからパスワードの入力を求められます。パスワードを入力すると、システムはパスワードを非表示にし、コマンドが正常に実行されたことと HSM に接続したことが出力 で示されます。

```
Enter password:

Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

Command:
```

Example: HSM からログアウトする

このコマンドは HSM からログアウトします。出力は、クラスターのすべての HSM からログアウトしたことを示しています。

```
Command: logoutHSM

Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

-h

このコマンドに関するヘルプを表示します。

-u

ログインユーザーのタイプを指定します。key_mgmt_util を使用するには、CU としてログインする必要があります。

必須: はい

-s

ログインユーザー名を指定します。

必須: はい

{-p |-hpswd}

-p でログインパスワードを指定します。パスワードは、入力するとプレーンテキストで表示されます。パスワードを非表示にするには、-hpswd の代わりにオプションの -p パラメータを使用して、プロンプトに従います。

必須: はい

関連トピック

- exit

KMU を使用して AWS CloudHSM キーの属性を設定する

AWS CloudHSM key_mgmt_util の setAttribute コマンドを使用して、現在のセッションでのみ有効なキーを、削除するまで存在する永続キーに変換します。この変換を行うために、キーのトークン属性 (OBJ_ATTR_TOKEN) の値を false (0) から true (1) に変更します。自分が所有するキーの属性のみ変更できます。

cloudhsm_mgmt_util の setAttribute コマンドを使用して、ラベルの変更、属性のラップ、アンラップ、暗号化、および復号化を行うこともできます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
setAttribute -h

setAttribute -o <object handle>
             -a 1
```

例

次の例では、セッションキーを永続キーに変換する方法を示します。

最初のコマンドは、genSymKey の -sess パラメーターを使用して、現在のセッションでのみ有効な 192 ビットの AES キーを作成します。出力は、新しいセッションキーのキーハンドルが 262154 であることを示しています。

```
Command: genSymKey -t 31 -s 24 -l tmpAES -sess

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262154

        Cluster Error Status
```

```
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

次のコマンドでは、findKey を使用して現在のセッションのセッションキーを確認します。出力は、キー 262154 がセッションキーであることを示しています。

```
Command:  findKey -sess 1

Total number of keys present 1

 number of keys matched from start index 0::0
262154

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

次のコマンドでは、setAttribute を使用してキー 262154 をセッションキーから永続キーに変換します。そのために、キーのトークン属性 (OBJ_ATTR_TOKEN) の値を 0 (false)から 1 (true)に変更します。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

このコマンドでは、-o パラメータを使用してキーハンドル (262154) を指定し、-a パラメータを使用してトークン属性を表す定数 (1) を指定します。コマンドを実行すると、トークン属性の値を指定するよう求められます。唯一の有効な値は 1 (true) です。これは、永続キーの値です。

```
Command: setAttribute -o 262154 -a 1
         This attribute is defined as a boolean value.
          Enter the boolean attribute value (0 or 1):1

        Cfm3SetAttribute returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

次のコマンドでは、262154 が永続キーになったことを確認するために、findKey を使用してセッションキー (-sess 1) と永続キー (-sess 0) を検索します。今回は、コマンドでセッションキーが検出されず、永続キーのリストで 262154 が返されます。

```
Command: findKey -sess 1

Total number of keys present 0

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS



Command: findKey -sess 0

Total number of keys present 5

 number of keys matched from start index 0::4
6, 7, 524296, 9, 262154

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-オ

ターゲットキーのキーハンドルを指定します。各コマンドに指定できるキーは 1 つのみです。キーのキーハンドルを取得するには、findKey を使用します。

必須: はい

-a

変更する属性を表す定数を指定します。唯一の有効な値は 1 です。これはトークン属性 OBJ_ATTR_TOKEN を表します。

属性とその整数値を取得するには、listAttributes を使用します。

必須: はい

関連トピック

- cloudhsm_mgmt_util の中で setAttribute です。
- getAttribute
- listAttributes
- キー属性リファレンス

KMU AWS CloudHSM を使用して署名を生成する

AWS CloudHSM key_mgmt_util の sign コマンドを使用して、選択したプライベートキーを使用してファイルの署名を生成します。

sign を使用するには、HSM 内にプライベートキーを持っておく必要があります。プライベートキーは、genSymKey コマンド、genRSAKeyPair コマンド、または genECCKeyPair コマンドで生成できます。また、importPrivateKey コマンドを使用してインポートすることもできます。詳細については、「キーの生成」を参照してください。

sign コマンドは、ユーザーが指定した (整数で表した) 署名機構を使って、メッセージファイルに署名します。使用できる署名機構のリストについては、「パラメータ」を参照してください。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
sign -h

sign -f <file name>
     -k <private key handle>
     -m <signature mechanism>
     -out <signed file name>
```

例

この例では、sign を使ってファイルに署名する方法を示します。

Example: ファイルに署名する

次のコマンドは、ハンドル 266309 を持つプライベートキーを使って messageFile という名前のファイルに署名します。このコマンドは、SHA256_RSA_PKCS (1) 署名機構を使い、署名したファイルを signedFile として保存します。

```
Command: sign -f messageFile -k 266309 -m 1 -out signedFile

Cfm3Sign returned: 0x00 : HSM Return: SUCCESS

signature is written to file signedFile

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-f

署名するファイルの名前。

必須: はい

-k

署名に使用するプライベートキーのハンドル。

必須: はい

-m

署名に使用する署名機構を表す整数。使用可能な署名機構は、次のような整数に対応します。

署名機構	対応する整数
SHA1_RSA_PKCS	0
SHA256_RSA_PKCS	1

署名機構	対応する整数
SHA384_RSA_PKCS	2
SHA512_RSA_PKCS	3
SHA224_RSA_PKCS	4
SHA1_RSA_PKCS_PSS	5
SHA256_RSA_PKCS_PSS	6
SHA384_RSA_PKCS_PSS	7
SHA512_RSA_PKCS_PSS	8
SHA224_RSA_PKCS_PSS	9
ECDSA_SHA1	15
ECDSA_SHA224	16
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

必須: はい

-out

署名したファイルの保存先とするファイルの名前。

必須: はい

関連トピック

- verify
- importPrivateKey
- genRSAKeyPair

- genECCKeyPair
- genSymKey
- キーの生成

KMU を使用して AWS CloudHSM キーをラップ解除する

AWS CloudHSM key_mgmt_util ツールの unWrapKey コマンドを使用して、ラップされた (暗号化された) 対称キーまたはプライベートキーをファイルから HSM にインポートします。key_mgmt_util の wrapKey コマンドでラップされた暗号化されたキーをインポートするように設計されていますが、他のツールでラップされたキーをアンラップするためにも使用できます。ただし、このような場合は、PKCS#11 または JCE ソフトウェアライブラリを使用して、キーをラップ解除することをお勧めします。

インポートされたキーは、 によって生成されたキーのように機能します AWS CloudHSM。ただし、OBJ_ATTR_LOCAL 属性の値は 0 であり、ローカルに生成されたものでないことを示しています。

キーをインポートしたら、必ずキーファイルをマークまたは削除してください。このコマンドでは、同じキーマテリアルを複数回インポートすることが禁止されません。その結果、異なるキーハンドルと同じキーマテリアルを持つ複数のキーにより、キーマテリアルの使用を追跡し、暗号化の制限を超えないようにすることが困難になります。

key_mgmt_util コマンドを実行する前に、 key_mgmt_util を起動し、Crypto User (CU) として HSM に ログインする 必要があります。

Syntax

```
unWrapKey -h

unWrapKey -f <key-file-name>
          -w <wrapping-key-handle>
          [-sess]
          [-min_srv <minimum-number-of-HSMs>]
          [-timeout <number-of-seconds>]
          [-aad <additional authenticated data filename>]
          [-tag_size <tag size>]
          [-iv_file <IV file>]
          [-attest]
          [-m <wrapping-mechanism>]
          [-t <hash-type>]
```

```
        [-nex]
        [-u <user id list>]
        [-m_value <number of users needed for approval>]
        [-noheader]
        [-l <key-label>]
        [-id <key-id>]
        [-kt <key-type>]
        [-kc <key-class>]
        [-i <unwrapping-IV>]
```

例

これらの例では、unWrapKey を使用して、ラップされたキーをファイルから HSM にインポートする方法を示します。最初の例では、wrapKey key_mgmt_util コマンドでラップされたキーをアンラップしているため、ヘッダーがあります。2 番目の例では、key_mgmt_util の外部でラップされたため、ヘッダーがないキーをアンラップします。

Example: キーのラップ解除 (ヘッダー付き)

このコマンドでは、3DES 対称キーのラップされたコピーを HSM にインポートします。キーはラベル 6 が付いた AES キーでラップ解除されます。このキーは、3 DES キーのラップに使用されたキーと暗号的に同一です。この出力は、ファイルのキーがラップ解除されてインポートされたことと、インポートされたキーのハンドルが 29 であることを示しています。

```
        Command:  unWrapKey -f 3DES.key -w 6 -m 4

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Key Unwrapped.  Key Handle: 29

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: キーのラップ解除 (ヘッダーなし)

このコマンドでは、3DES 対称キーのラップされたコピーを HSM にインポートします。キーはラベル 6 が付いた AES キーでラップ解除されます。このキーは、3 DES キーのラップに使用されたキーと暗号的に同一です。この 3DES キーは key_mgmt_util でラップされていないため、noheader パラメータは、キー・ラベル (unwrapped3DES)、キー・クラス (4)、キー・タイプ (21) など必要な付

随パラメータとともに指定されます。この出力は、ファイルのキーがラップ解除されてインポートされたことと、インポートされたキーのハンドルが 8 であることを示しています。

```
Command: unWrapKey -f 3DES.key -w 6 -noheader -l unwrapped3DES -kc 4 -kt 21 -m 4

      Cfm3CreateUnwrapTemplate2 returned: 0x00 : HSM Return: SUCCESS
       Cfm2UnWrapWithTemplate3 returned: 0x00 : HSM Return: SUCCESS

       Key Unwrapped. Key Handle: 8

       Cluster Error Status
       Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-f

ラップされたキーが含まれているファイルのパスと名前を指定します。

必須: はい

-w

ラップキーを指定します。HSM の AES キーまたは RSA キーのキーハンドルを入力します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

ラッピングキーを作成するには、genSymKey を使用して AES キー (タイプ 31) を生成するか、genRSAKeyPair を使用して RSA キーペア (タイプ 0) を生成します。RSA キーペアを使用している場合は、必ず一方のキーでキーをラップし、もう一方のキーでアンラップしてください。キーをラッピングキーとして使用できることを確認するには、getAttribute を使用して、定数 OBJ_ATTR_WRAP で表される 262 属性の値を取得します。

必須: はい

-sess

現在のセッションにのみ存在するキーを作成します。セッション終了後、キーをリカバリすることはできません。

このパラメータは、別のキーを暗号化してからすばやく復号化するラッピングキーなど、キーが短時間だけ必要な場合に使用します。セッション終了後に復号する必要がある可能性のあるデータを暗号化するためにセッションキーを使用しないでください。

セッションキーを永続(トークン)キーに変更するには、setAttribute を使用します。

デフォルト: キーは永続的です。

必須: いいえ

-min_srv

`-timeout` パラメーターの値が期限切れになる前に、キーが同期される HSM の最小数を指定します。キーが割り当てられた時間内に指定された数のサーバーに同期されない場合は、作成されません。

AWS CloudHSM は、すべてのキーをクラスター内のすべての HSM に自動的に同期します。プロセスを高速化するため、`min_srv` の値をクラスターの HSM の数より少なく設定し、低いタイムアウト値を設定します。ただし、一部のリクエストでキーが生成されない場合があることに注意してください。

デフォルト: 1

必須: いいえ

-timeout

キーが `min_srv` パラメータで指定された HSM の数に同期されるのをコマンドが待機する時間 (秒単位) を指定します。

このパラメータは、`min_srv` パラメータがコマンドでも使用されている場合にのみ有効です。

デフォルト: タイムアウトなし。このコマンドは無期限に待機し、キーが最小数のサーバーと同期されている場合にのみ戻ります。

必須: いいえ

-attest

クラスターを実行するファームウェアが改ざんされていないことを確認する整合性チェックを実行します。

デフォルト: 認証チェックなし。

必須: いいえ

-nex

キーを抽出できなくなります。生成されたキーは HSM からエクスポートできません。

デフォルト: キーは抽出可能です。

必須: いいえ

-m

ラップメカニズムを表す値。CloudHSM は、次のメカニズムをサポートしています。

メカニズム	値
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (最大データサイズについてはこのセクションの後半のメモを参照)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (最大データサイズについてはこのセクションの後半のメモを参照)。今後の変更については、以下の注記「1」を参照してください。	12

必須: はい

> **Note**
>
> RSA_OAEP ラップメカニズムを使用する場合、ラップ可能な最大キーサイズは、次のように、RSA キーのモジュールと、指定したハッシュの長さによって決まります: 最大キーサイズ = modulusLengthInBytes-(2*hashLengthInBytes)-2。

RSA_PKCS ラップメカニズムを使用する場合、ラップ可能な最大キーサイズは、RSA キーのモジュールによって次のように決まります: 最大キーサイズ = (modulusLengthInBytes -11)。

-t

ハッシュアルゴリズム	値
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (RSA_AES および RSA_OAEP メカニズムに対して有効)	6

必須: いいえ

-noheader

key_mgmt_util の外部でラップされたキーをラップ解除する場合は、このパラメータと他のすべての関連パラメータを指定する必要があります。

必須: いいえ

> **Note**
> このパラメータを指定する場合は、-noheader パラメータも指定する 必要があります。

- -l

  ラップ解除されたキーに追加するラベルを指定します。

  必須: はい

- -kc

ラップ解除するキーのクラスを指定します。使用できる値は以下のとおりです。

3 = パブリックキーとプライベートキーのペアのプライベートキー

4: シークレット (対称) キー

必須: はい

- -kt

  ラップ解除するキーのタイプを指定します。使用できる値は以下のとおりです。

  0 = RSA

  1 = DSA

  3 = ECC

  16 = GENERIC_SECRET

  21 = DES3

  31 = AES

  必須: はい

オプションで次の -noheader パラメータを指定することもできます。

- -id

  ラップ解除されたキーに追加する ID。

  必須: いいえ

- -i

  使用するラップ解除対象の初期ベクトル (IV)。

  必須: いいえ

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

関連トピック

- wrapKey
- exSymKey
- imSymKey

KMU AWS CloudHSMを使用してファイルの署名を検証する

AWS CloudHSM key_mgmt_util の verify コマンドを使用して、ファイルが特定のキーによって署名されているかどうかを確認します。その際、verify コマンドは、署名されたファイルをソースファイルと比較し、両ファイルが特定のパブリックキーと署名機構に基づいて暗号的に関連するかどうかを分析します。ファイルは signオペレーション AWS CloudHSM でサインインできます。

署名機構は、「パラメータ」セクションにリストされている整数によって表されます。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
verify -h

verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

例

これらの例は、verify を使って、特定のファイルの署名に特定のパブリックキーが使用されたかどうかを確認する方法を示します。

Example: ファイル署名の認証

このコマンドは、hardwarCert.crt というファイルがパブリックキー 262276 と署名機構 SHA256_RSA_PKCS を使って署名され、hardwareCertSigned という署名ファイルが作成されたかどうかの検証を試みます。指定のパラメータが真の署名関係を表すため、コマンドは、成功メッセージを返します。

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1
```

```
Signature verification successful

Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example: 偽の署名関係を証明する

このコマンドは、hardwareCert.crt というファイルがパブリックキー 262276 と署名機構 SHA256_RSA_PKCS を使って署名され、userCertSigned という署名ファイルが作成されたことを検証します。指定のパラメータが真の署名関係を構成していないため、コマンドは、エラーメッセージを返します。

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
Cfm3Verify returned: 0x1b

CSP Error: ERR_BAD_PKCS_DATA
```

パラメータ

このコマンドでは、以下のパラメータを使用します。

-f

元のメッセージファイルの名前。

必須: はい

-s

署名したファイルの名前。

必須: はい

-k

ファイルの署名に使用されたと考えられるパブリックキーのハンドル。

必須: はい

-m

ファイルの署名に使用するよう提案された署名機構を表す整数。使用可能な署名機構は、次のような整数に対応します。

署名機構	対応する整数
SHA1_RSA_PKCS	0
SHA256_RSA_PKCS	1
SHA384_RSA_PKCS	2
SHA512_RSA_PKCS	3
SHA224_RSA_PKCS	4
SHA1_RSA_PKCS_PSS	5
SHA256_RSA_PKCS_PSS	6
SHA384_RSA_PKCS_PSS	7
SHA512_RSA_PKCS_PSS	8
SHA224_RSA_PKCS_PSS	9
ECDSA_SHA1	15
ECDSA_SHA224	16
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

必須: はい

関連トピック

- sign
- getCert
- キーの生成

KMU を使用して AWS CloudHSM キーをエクスポートする

AWS CloudHSM key_mgmt_util の wrapKey コマンドを使用して、対称キーまたはプライベートキーの暗号化されたコピーをハードウェアセキュリティモジュール (HSM) からファイルにエクスポートします。wrapKey を実行するときに、エクスポートするキー、エクスポートするキーを暗号化 (ラップ) するための HSM 上のキー、出力ファイルを指定します。

wrapKey コマンドは、暗号化されたキーを指定したファイルに書き込みますが、キーを HSM から削除したり、暗号化オペレーションでのキーの使用を禁止したりすることはありません。同じキーを複数回エクスポートできます。

キーは、キーの所有者 (キーを作成した Crypto User (CU)) のみエクスポートできます。キーを共有するユーザーは、キーを暗号化オペレーションで使用することはできますが、エクスポートすることはできません。

暗号化されたキーを HSM に再度インポートするには、unWrapKey を使用します。HSM からプレーンテキストキーをエクスポートするには、必要に応じて exSymKey または exportPrivateKey を使用します。aesWrapUnwrap コマンドは、wrapKey で暗号化したキーを復号 (ラップ解除) することはできません。

key_mgmt_util コマンドを実行する前に、key_mgmt_util を起動し、Crypto User (CU) として HSM にログインする 必要があります。

Syntax

```
wrapKey -h

wrapKey -k <exported-key-handle>
        -w <wrapping-key-handle>
        -out <output-file>
        [-m <wrapping-mechanism>]
        [-aad <additional authenticated data filename>]
        [-t <hash-type>]
        [-noheader]
        [-i <wrapping IV>]
        [-iv_file <IV file>]
        [-tag_size <num_tag_bytes>>]
```

例

Example

次のコマンドでは、192 ビット Triple DES (3DES) 対称キー (キーハンドル 7) をエクスポートします。HSM で 256 ビット AES キー (キーハンドル 14) を使用してキー 7 をラップします。次に、暗号化された 3DES キーを 3DES-encrypted.key ファイルに書き込みます。

次の出力は、キー 7 (3DES キー) が正常にラップされて指定したファイルに書き込まれたことを示しています。暗号化されたキーの長さは 307 バイトです。

```
        Command:  wrapKey -k 7 -w 14 -out 3DES-encrypted.key -m 4

        Key Wrapped.

        Wrapped Key written to file "3DES-encrypted.key length 307

        Cfm2WrapKey returned: 0x00 : HSM Return: SUCCESS
```

パラメータ

-h

コマンドに関するヘルプを表示します。

必須: はい

-k

エクスポートするキーのキーハンドル。所有する対称キーまたはプライベートキーのキーハンドルを入力します。キーハンドルを見つけるには、findKey コマンドを使用します。

キーがエクスポート可能であることを検証するには、getAttribute コマンドを使用して、OBJ_ATTR_EXTRACTABLE 属性の値を取得します。この属性は定数 354 で表されます。キー属性の解釈については、AWS CloudHSM KMU のキー属性リファレンス を参照してください。

ユーザーが所有するキーのみをエクスポートすることができます。キーの所有者を確認するには、getKeyInfo コマンドを使用します。

必須: はい

-w

ラップキーを指定します。HSM の AES キーまたは RSA キーのキーハンドルを入力します。このパラメータは必須です。キーハンドルを見つけるには、findKey コマンドを使用します。

ラッピングキーを作成するには、genSymKey を使用して AES キー (タイプ 31) を生成するか、genRSAKeyPair を使用して RSA キーペア (タイプ 0) を生成します。RSA キーペアを使用している場合は、必ず一方のキーでキーをラップし、もう一方のキーでアンラップしてください。キーをラッピングキーとして使用できることを確認するには、getAttribute を使用して、定数 OBJ_ATTR_WRAP で表される 262 属性の値を取得します。

必須: はい

-out

出力ファイルのパスと名前。コマンドが成功すると、このファイルに、エクスポートされたキーの暗号化されたコピーが格納されます。既存のファイルがある場合は、警告なしに上書きされます。

必須: はい

-m

ラップメカニズムを表す値。CloudHSM は、次のメカニズムをサポートしています。

メカニズム	値
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (最大データサイズについてはこのセクションの後半のメモを参照)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11

メカニズム	値
RSA_PKCS (最大データサイズについてはこのセクションの後半のメモを参照)。今後の変更については、以下の注記「1」を参照してください。	12

必須: はい

> **Note**
>
> RSA_OAEP ラップメカニズムを使用する場合、ラップ可能な最大キーサイズは、次のように、RSA キーのモジュールと、指定したハッシュの長さによって決まります: 最大キーサイズ = (modulusLengthInBytes-2*hashLengthInBytes-2)。
> RSA_PKCS ラップメカニズムを使用する場合、ラップ可能な最大キーサイズは、RSA キーのモジュールによって次のように決まります: 最大キーサイズ = (modulusLengthInBytes -11)。

-t

ハッシュアルゴリズムを表す値。CloudHSM は、次のアルゴリズムをサポートしています。

ハッシュアルゴリズム	値
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (RSA_AES および RSA_OAEP メカニズムに対して有効)	6

必須: いいえ

-aad

AAD を含むファイル名。

> **Note**
>
> AES_GCM および CLOUDHSM_AES_GCM メカニズムに対してのみ有効です。

必須: いいえ

-noheader

CloudHSM 固有の キー属性を指定するヘッダーを除外します。このパラメータは、key_mgmt_util 以外のツールでキーをアンラップする場合に のみ 使用してください。

必須: いいえ

-i

初期化ベクトル (IV) (16 進値)。

> **Note**
>
> CLOUDHSM_AES_KEY_WRAP と NIST_AES_WRAP メカニズムの -noheader パラメータで渡された場合にのみ有効です。

必須: いいえ

-iv_file

応答で取得した IV 値を書き込むファイル。

> **Note**
>
> AES_GCM メカニズムの -noheader パラメータで渡された場合にのみ有効です。

必須: いいえ

-tag_size

ラップされた blob とともに保存されるタグのサイズ。

> **Note**
>
> AES_GCM と CLOUDHSM_AES_GCM メカニズムの -noheader パラメータで渡された場合にのみ有効です。タグの最小サイズは 8 です。

必須: いいえ

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

関連トピック

- exSymKey
- imSymKey
- unWrapKey

AWS CloudHSM KMU のキー属性リファレンス

AWS CloudHSM key_mgmt_util コマンドは定数を使用して、ハードウェアセキュリティモジュール (HSM) 内のキーの属性を表します。このトピックは、属性の識別、コマンドでその属性を表す定数の検索、およびその値の理解に役立ちます。

キーの作成時に、キー属性を設定します。キーが永続的であるかセッションにのみ存在するかを示すトークン属性を変更するには、key_mgmt_utilの setAttribute コマンドを使用します。ラベルを変更、属性をラップ、アンラップ、暗号化、または復号化するには、cloudhsm_mgmt_util の setAttributeコマンドを使用します。

属性とその定数のリストを取得するには、listAttributes を使用します。キーの属性値を取得するには、getAttribute を使用します。

次の表に、キーの属性、その定数、および有効な値を示します。

属性	定数	値
OBJ_ATTR_ALL	512	すべての属性を表します。

属性	定数	値
OBJ_ATTR_ALWAYS_SENSITIVE	357	0: False。 1: True。
OBJ_ATTR_CLASS	0	2 : 公開 - プライベートキーのキーペアの公開キー。 3 : 公開 - プライベートキーのキーペアの公開キー。 4: シークレット (対称) キー。
OBJ_ATTR_DECRYPT	261	0: False。 1: True。キーはデータの復号に使用できます。
OBJ_ATTR_DERIVE	268	0: False。 1: True。この関数は、キーを派生させます。
OBJ_ATTR_DESTROYABLE	370	0: False。 1: True。
OBJ_ATTR_ENCRYPT	260	0: False。 1: True。キーはデータの暗号化に使用できます。
OBJ_ATTR_EXTRACTABLE	354	0: False。 1: True。キーは HSM からエクスポートできます。
OBJ_ATTR_ID	258	ユーザー定義の文字列。クラスター内で一意である必要があります。デフォルトは空の文字列です。

属性	定数	値
OBJ_ATTR_KCV	371	キーのキーチェック値。詳細については、「その他の詳細」を参照してください。
OBJ_ATTR_KEY_TYPE	256	0: RSA。 1: DSA。 3: EC。 16: 汎用秘密。 18: RC4。 21: Triple DES (3DES)。 31: AES。
OBJ_ATTR_LABEL	3	ユーザー定義の文字列。クラスター内で一意である必要はありません。
OBJ_ATTR_LOCAL	355	0. False。キーは HSM にインポートされました。 1: True。
OBJ_ATTR_MODULUS	288	RSA キーペアを生成するために使用されたモジュラス。EC キーの場合、この値は ANSI X9.62 ECPoint 値「Q」の DER エンコーディングを 16 進数形式で表します。 他のキータイプの場合、この属性は存在しません。

属性	定数	値
OBJ_ATTR_MODULUS_BITS	289	RSA キーペアを生成するために使用されたモジュラスの長さ。EC キーの場合、これはキーの生成に使用された楕円曲線の ID を表します。 他のキータイプの場合、この属性は存在しません。
OBJ_ATTR_NEVER_EXTRACTABLE	356	0: False。 1: True。このキーは HSM からエクスポートできません。
OBJ_ATTR_PUBLIC_EXPONENT	290	RSA キーペアを生成するために使用された公開指数。 他のキータイプの場合、この属性は存在しません。
OBJ_ATTR_PRIVATE	2	0: False。 1: True。この属性は、認証されていないユーザーがキーの属性を表示できるかどうかを示します。CloudHSM PKCS#11 プロバイダーでは、現在パブリックセッションはサポートされていないため、すべてのキー (パブリックキーとプライベートキーのペアのパブリックキーを含む) のこの属性は 1 に設定されています。

属性	定数	値
OBJ_ATTR_SENSITIVE	259	0: False。公開 - プライベートキーのキーペアの公開キー。 1: True。
OBJ_ATTR_SIGN	264	0: False。 1: True。キーは署名 (プライベートキー) に使用できます。
OBJ_ATTR_TOKEN	1	0: False。セッションキー。 1: True。永続キー。
OBJ_ATTR_TRUSTED	134	0: False。 1: True。
OBJ_ATTR_UNWRAP	263	0: False。 1: True。キーはキーの復号に使用できます。
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	値は、このラッピングキーを使用してラップ解除されたキーに適用される属性テンプレートを使用する必要があります。
OBJ_ATTR_VALUE_LEN	353	キーの長さ (バイト単位)
OBJ_ATTR_VERIFY	266	0: False。 1: True。キーは検証 (パブリックキー) に使用できます。

属性	定数	値
OBJ_ATTR_WRAP	262	0: False。 1: True。キーはキーの暗号化に使用できます。
OBJ_ATTR_WRAP_TEMPLATE	1073742353	値は、属性テンプレートを使用し、このラッピングキーでラップされたキーと一致させる必要があります。
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0: False。 1: True。

その他の詳細

キーチェック値 (KCV)。

キーチェック値 (KCV) は、HSM がキーをインポートまたは生成するときに生成されるキーの 3 バイトのハッシュまたはチェックサムです。キーを エクスポートした後など、HSM の外部で KCV を計算することもできます。次に、KCV 値を比較して、キーのアイデンティティと整合性を確認できます。キーの KCV を取得するには、getAttribute を使用します。

AWS CloudHSM は、次の標準メソッドを使用してキーチェック値を生成します。

- 対称キー: ゼロブロックをキーで暗号化した結果の最初の 3 バイト。
- 非対称キーペア: 公開キーの SHA-1 ハッシュの最初の 3 バイト。
- HMAC キー: 現時点では HMAC キーの KCV はサポートされていません。

AWS CloudHSM クライアント SDKsオフロード

クライアント SDK を使用して、暗号化オペレーションをプラットフォームまたは言語ベースのアプリケーションからハードウェアセキュリティモジュール (HSM) にオフロードします。

AWS CloudHSM には 2 つのメジャーバージョンがあり、クライアント SDK 5 が最新です。クライアント SDK 3 (以前のシリーズ) よりも多くの、さまざまな利点があります。詳細については、「Benefits of クライアント SDK 5」を参照してください。サポートされるプラットフォームの詳細については、「AWS CloudHSM クライアント SDK 5 がサポートするプラットフォーム」を参照してください。

以下のトピックでは、 AWS CloudHSM クライアント SDKsの操作方法について説明します。

AWS CloudHSM は、次のコンポーネントをサポートしています。

the section called “PKCS #11 ライブラリ”

PKCS #11 は、ハードウェアセキュリティモジュール (HSM) で暗号化オペレーションを実行するための標準です。 AWS CloudHSM は、PKCS #11 バージョン 2.40 に準拠した PKCS #11 ライブラリの実装を提供します。

the section called “OpenSSL Dynamic Engine”

AWS CloudHSM OpenSSL Dynamic Engine を使用すると、OpenSSL API を使用して暗号化オペレーションを CloudHSM OpenSSL クラスターにオフロードできます。

the section called “OpenSSL プロバイダー”

AWS CloudHSM OpenSSL Dynamic Engine Provider を使用すると、OpenSSL Provider API を使用して暗号化オペレーションを CloudHSM OpenSSL クラスターにオフロードできます。

the section called “JCE プロバイダー”

AWS CloudHSM JCE プロバイダーは Java 暗号化アーキテクチャ (JCA) に準拠しています。そのプロバイダーは HSM 上での暗号化オペレーションを許可します。

the section called “キーストレージプロバイダー (KSP)”

Windows 用の AWS CloudHSM クライアントには、CNG プロバイダーと KSP プロバイダーが含まれています。

トピック

- AWS CloudHSM クライアント SDK のバージョンを確認する

- AWS CloudHSM クライアント SDK コンポーネントのサポートを比較する
- AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行
- クライアント SDK 5 を使用して を操作する AWS CloudHSM
- 以前の SDK バージョンを使用して を操作する AWS CloudHSM

AWS CloudHSM クライアント SDK のバージョンを確認する

次のコマンドを使用して、 AWS CloudHSMで使用しているクライアント SDKのバージョンを確認します。

Amazon Linux

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

以下のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

以下のコマンドを使用します。

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

以下のコマンドを使用します。

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

以下のコマンドを使用します。

```
apt list --installed | grep ^cloudhsm
```

Windows Server

以下のコマンドを使用します。

```
wmic product get name,version
```

AWS CloudHSM クライアント SDK コンポーネントのサポートを比較する

クライアント SDK 3 には、コマンドラインツールに加え、さまざまなプラットフォームまたは言語ベースのアプリケーションから HSM に暗号化オペレーションをオフロードできるコンポーネントが含まれています。クライアント SDK 5 は、クライアントSDK 3と同等の機能を持ちますが、CNGとKSPプロバイダーはまだサポートしていません。次の表では、クライアント SDK 3 と クライアント SDK 5 のコンポーネントの可用性を比較します。

コンポーネント	クライアント SDK 5	クライアント SDK 3
PKCS #11 ライブラリ	はい	はい
JCE プロバイダー	はい	はい
OpenSSL Dynamic Engine	はい	はい
キーストレージプロバイダー (KSP)	はい	はい
CloudHSM 管理ユーティリティ (CMU)[1]	はい	はい
キー管理ユーティリティ (KMU)[1]	はい	はい
設定ツール	はい	はい

[1] CMU と KMU コンポーネントは、クライアント SDK 5 を搭載した CloudHSM CLI に含まれています。

次のセクションでは、コンポーネントについて説明します。

PKCS #11 ライブラリ

PKCS #11 は、ハードウェアセキュリティモジュール (HSMs) で暗号化オペレーションを実行するための標準です。 は、PKCS #11 バージョン 2.40 に準拠した PKCS #11 ライブラリの実装 AWS CloudHSM を提供します。

- クライアント SDK 3 の場合、PKCS #11 ライブラリは Linux の基本サポートに一致する Linux のみのコンポーネントです。詳細については、「the section called “クライアント SDK 3 の Linux サポート”」を参照してください。
- クライアント SDK 5 の場合、PKCS #11 ライブラリは、Linux および Windows クライアント SDK 5 の基本サポートに一致するクロスプラットフォームコンポーネントです。詳細については、「the section called “クライアント SDK 5 の Linux サポート”」および「the section called “クライアント SDK 5 の Windows サポート”」を参照してください。

CloudHSM 管理ユーティリティ (CMU)

CloudHSM 管理ユーティリティ (CMU) コマンド ライン ツールは、Crypto Officer が HSM 内のユーザーを管理するのに役立ちます。これには、ユーザーの作成、削除および一覧表示とユーザーパスワードの変更を行うツールが含まれています。詳細については、「AWS CloudHSM 管理ユーティリティ (CMU)」を参照してください。

キー管理ユーティリティ (KMU)

キー管理ユーティリティ (KMU) は、Crypto User (CU) がハードウェアセキュリティモジュール (HSM) のキーを管理するのに役立つコマンドラインツールです。詳細については、「AWS CloudHSM キー管理ユーティリティ (KMU)」を参照してください。

JCE プロバイダー

AWS CloudHSM JCE プロバイダーは Java 暗号化アーキテクチャ (JCA) に準拠しています。そのプロバイダーは HSM 上での暗号化オペレーションを許可します。

JCE プロバイダーは Linux の基本サポートに一致する Linux のみのコンポーネントです。詳細については、「the section called “クライアント SDK 3 の Linux サポート”」を参照してください。

- クライアント SDK 3 には OpenJDK 1.8 が必要です

OpenSSL Dynamic Engine

AWS CloudHSM OpenSSL Dynamic Engine を使用すると、OpenSSL API を使用して暗号化オペレーションを CloudHSM OpenSSL クラスターにオフロードできます。

- クライアント SDK 3 の場合、OpenSSL Dynamic Engine は Linux の基本サポートと一致 しない Linux のみのコンポーネントです。以下の除外項目を参照してください。

- OpenSSL 1.0.2[f+] が必要です。

サポートされていないプラットフォーム

- CentOS 8
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 18.04 LTS

これらのプラットフォームは、クライアント SDK 3 の OpenSSL Dynamic Engine と互換性がないバージョンの OpenSSL が付属しています。 AWS CloudHSM は、クライアント SDK 5 の OpenSSL Dynamic Engine でこれらのプラットフォームをサポートしています。

- クライアント SDK 5 の場合、OpenSSL Dynamic Engine は Linux 専用のコンポーネントで、OpenSSL 1.0.2、1.1.1、または 3.x が必要です。

キーストレージプロバイダー (KSP)

キーストレージプロバイダー (KSP) は、Microsoft Windows オペレーティングシステム固有の暗号 API です。

クライアント SDK 3 では、CNG および KSP プロバイダーは、Windows の基本サポートに一致する Windows のみのコンポーネントです。詳細については、「AWS CloudHSM クライアント SDK 3 の Windows サポート」を参照してください。

クライアント SDK 5 では、キーストレージプロバイダー (KSP) プロバイダーは、Windows の基本サポートに一致する Windows のみのコンポーネントです。詳細については、「AWS CloudHSM クライアント SDK 5 の Windows サポート」を参照してください。

AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行

では AWS CloudHSM、顧客アプリケーションは AWS CloudHSM クライアントソフトウェア開発キット (SDK) を使用して暗号化オペレーションを実行します。クライアント SDK 5 は、新しい機能とプラットフォームサポートが継続的に追加される、主要な SDK です。

クライアント SDK 3 には、ユーザー管理用の CMU と、キー管理およびキー操作用の KMU という 2 つの独立したコマンドラインツールが含まれています。クライアント SDK 5 は、CMU と KMU (クライアント SDK 3 で提供されていたツール) の機能を 1 つのツール AWS CloudHSM コマンドラインインターフェイス (CLI) に統合します。ユーザー管理オペレーションは、 サブコマンド

CloudHSM CLI のユーザーカテゴリ と CloudHSM CLI のクォーラムカテゴリ で実行できます。キー管理オペレーションは key サブコマンド、暗号化オペレーションは crypto サブコマンドの下にあります。コマンドの完全なリストについては、「CloudHSM CLI コマンドのリファレンス」を参照してください。

移行の利点については、「AWS CloudHSM クライアント SDK 5 の利点」を参照してください。

クライアント SDK 3 からクライアント SDK 5 への移行手順については、以下のトピックを参照してください。 AWS CloudHSM Client SDK の最新バージョンは 5.16 です。

- AWS CloudHSM PKCS #11 ライブラリをクライアント SDK 3 からクライアント SDK 5 に移行する
- OpenSSL 動的エンジンを AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行する
- キーストレージプロバイダー (KSP) を AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行する
- JCE プロバイダーを AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行する

CloudHSM CLI でサポートされていない機能やユースケースについては、AWS サポート にお問い合わせください。

AWS CloudHSM PKCS #11 ライブラリをクライアント SDK 3 からクライアント SDK 5 に移行する

このトピックを使用して、 AWS CloudHSM PKCS #11 ライブラリをクライアント SDK 3 からクライアント SDK 5 に移行します。移行の利点については、「AWS CloudHSM クライアント SDK 5 の利点」を参照してください。

では AWS CloudHSM、顧客アプリケーションは AWS CloudHSM クライアントソフトウェア開発キット (SDK) を使用して暗号化オペレーションを実行します。クライアント SDK 5 は、新しい機能とプラットフォームサポートが継続的に追加される、主要な SDK です。

すべてのプロバイダーの移行手順を確認するには、「」を参照してくださいAWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行。

重大な変更に対処して準備する

これらの重大な変更を確認し、それに応じて開発環境でアプリケーションを更新します。

ラップメカニズムが変更されました

クライアント SDK 3 メカニズム	同等のクライアント SDK 5 メカニズム
CKM_AES_KEY_WRAP	CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
CKM_AES_KEY_WRAP_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

ECDH

クライアント SDK 3 では、ECDH を使用して KDF を指定できます。この機能は、現在クライアント SDK 5 では利用できません。アプリケーションにこの機能が必要な場合は、サポートにお問い合わせください。

キーハンドルがセッション固有になりました

クライアント SDK 5 でキーハンドルを正常に使用する場合、アプリケーションを実行するたびにキーハンドルを取得する必要があります。異なるセッション間で同じキーハンドルを使用することが予想される既存のアプリケーションがある場合、アプリケーションを実行するたびにキーハンドルを取得するようにコードを変更することが必要です。キーハンドルの取得については、この AWS CloudHSM PKCS #11 の例を参照してください。この変更は、PKCS #11 2.40 仕様 に準拠したものです。

クライアント SDK 5 への移行

このセクションの指示に従って、クライアント SDK 3 から クライアント SDK 5 に移行します。

> **Note**
>
> Amazon Linux、Ubuntu 16.04、Ubuntu 18.04、CentOS 6、CentOS 8、および RHEL 6 は現在、クライアント SDK 5 ではサポートされていません。現在、クライアント SDK 3 でこれらのプラットフォームのいずれかを使用している場合は、クライアント SDK 5 に移行するときに別のプラットフォームを選択する必要があります。

1. クライアント SDK 3 用の PKCS #11 ライブラリのアンインストールします。

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client-pkcs11
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client-pkcs11
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client-pkcs11
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-client-pkcs11
   ```

   Ubuntu 16.04 LTS

   ```
   $ sudo apt remove cloudhsm-client-pkcs11
   ```

   Ubuntu 18.04 LTS

   ```
   $ sudo apt remove cloudhsm-client-pkcs11
   ```

2. クライアント SDK 3 の Client Daemon を停止します。

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

3. クライアント SDK 3 の Client Daemon をアンインストールします。

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client
```

> **Note**
>
> カスタム設定を再度有効にする必要があります。

4. 「AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリをインストールする」の手順に従って、クライアント SDK PKCS #11 ライブラリをインストールします。
5. クライアント SDK 5 では、新しい設定ファイル形式とコマンドラインブートストラップツールが導入されました。クライアント SDK 5 PKCS #11 ライブラリをブートストラップするには、クライアント SDK をブートストラップする のユーザーガイドに記載されている手順に従ってください。
6. アプリケーションは開発環境で実行されていること。既存のコードを更新して、最終的な移行前に重大な変更を解決します。

関連トピック

- のベストプラクティス AWS CloudHSM

OpenSSL 動的エンジンを AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行する

このトピックを使用して、OpenSSL Dynamic Engine を AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行します。移行の利点については、「AWS CloudHSM クライアント SDK 5 の利点」を参照してください。

では AWS CloudHSM、顧客アプリケーションは AWS CloudHSM クライアントソフトウェア開発キット (SDK) を使用して暗号化オペレーションを実行します。クライアント SDK 5 は、新しい機能とプラットフォームサポートが継続的に追加される、主要な SDK です。

> **Note**
>
> 乱数生成は現在、OpenSSL Dynamic Engine を使用するクライアント SDK 5 ではサポートされていません。

すべてのプロバイダーの移行手順を確認するには、「」を参照してくださいAWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行。

クライアント SDK 5 への移行

このセクションの指示に従って、クライアント SDK 3 から クライアント SDK 5 に移行します。

> **Note**
>
> Amazon Linux、Ubuntu 16.04、Ubuntu 18.04、CentOS 6、CentOS 8、および RHEL 6 は現在、クライアント SDK 5 ではサポートされていません。現在、クライアント SDK 3 でこれらのプラットフォームのいずれかを使用している場合は、クライアント SDK 5 に移行するときに別のプラットフォームを選択する必要があります。

1. クライアント SDK 3 用の OpenSSL Dynamic Engine をアンインストールする

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client-dyn
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client-dyn
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client-dyn
   ```

RHEL 8

```
$ sudo yum remove cloudhsm-client-dyn
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client-dyn
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client-dyn
```

2. クライアント SDK 3 の Client Daemon を停止します。

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

3. クライアント SDK 3 の Client Daemon をアンインストールします。

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-client
   ```

   Ubuntu 16.04 LTS

   ```
   $ sudo apt remove cloudhsm-client
   ```

   Ubuntu 18.04 LTS

   ```
   $ sudo apt remove cloudhsm-client
   ```

   > **Note**
   >
   > カスタム設定を再度有効にする必要があります。

4. 「AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine をインストールする」の手順に従って、クライアント SDK OpenSSL Dynamic Engine をインストールします。
5. クライアント SDK 5 では、新しい設定ファイル形式とコマンドラインブートストラップツールが導入されました。クライアント SDK 5 OpenSSL Dynamic Engine をブートストラップするには、ユーザー ガイドの クライアント SDK をブートストラップする に記載されている手順に従ってください。

6. アプリケーションは開発環境で実行されていること。既存のコードを更新して、最終的な移行前に重大な変更を解決します。

関連トピック

- のベストプラクティス AWS CloudHSM

キーストレージプロバイダー (KSP) を AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行する

このトピックでは、 AWS CloudHSM クライアント SDK 3 から クライアント SDK 5 へキーストレージプロバイダー (KSP) を移行する方法について説明します。 AWS CloudHSM Client SDK の最新バージョンは 5.16 です。移行のメリットに関する詳細については、AWS CloudHSM クライアント SDK 5 の利点 を参照してください。

では AWS CloudHSM、 AWS CloudHSM クライアントソフトウェア開発キット (SDK) を使用して暗号化オペレーションを実行します。クライアント SDK 5 は、新機能やプラットフォームサポートの更新を受け取る主要な SDK です。

すべてのプロバイダーの移行手順については、AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行 を参照してください。

クライアント SDK 5 への移行

1. クライアント SDK 3 の Client Daemon を停止します。

```
PS C:\> Stop-Service "AWS CloudHSM Client"
```

2. Windows Server インスタンスにクライアント SDK 5 キーストレージプロバイダー (KSP) をインストールします。手順については、「AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) をインストールする」を参照してください。
3. 新しい設定ファイル形式およびコマンドラインのブートストラップツールを使用して、クライアント SDK 5 キーストレージプロバイダー (KSP) を設定します。手順については、「クライアント SDK をブートストラップする」を参照してください。
4. AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) には、SDK3 で生成されたキーリファレンスファイルをサポートする SDK3 互換性モードが含まれていま

す。詳細については、「のキーストレージプロバイダー (KSP) の SDK3 互換性モード AWS CloudHSM」を参照してください。

> **Note**
>
> クライアント SDK 5 でクライアント SDK 3 によって生成されたキー参照ファイルを使用する場合、SDK3 互換モードを有効にする必要があります。

新しい Windows Server インスタンスへの移行

1. 新しい Windows Server インスタンス上で、クライアント SDK 5 への移行の全手順を完了します。
2. 既存のキー参照ファイルを確認する

   元の Windows Server インスタンスで、`C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition` にキー参照ファイルが存在するか確認します。

   - キー参照ファイルが存在する場合は、`GlobalPartition` を含む `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP` のすべてのコンテンツを新しい Windows Server インスタンスの同じディレクトリパスにコピーします。このディレクトリが存在しない場合は、作成します。
   - キー参照ファイルが存在しない場合は、新しい Windows Server インスタンスで `cloudhsm-cli key generate-file --encoding ksp-key-reference` を使用して作成します。手順については、「KSP キー参照を生成します (Windows)」を参照してください。
3. ルート証明書を検証する

   信頼されたルート証明機関ストア内のルート証明書を確認します。

```
PS C:\Users\Administrator\Desktop> certutil -store Root

Root "Trusted Root Certification Authorities"
================ Certificate 0 ================
Serial Number: certificate-serial-number
Issuer: CN=MYRootCA
 NotBefore: 2/5/2020 1:38 PM
 NotAfter: 2/5/2021 1:48 PM
 Issuer: CN=MYRootCA
```

```
Signature matches Public Key
Root Certificate: Subject matches Issuer
Cert Hash(sha1): cert-hash
No key provider information
Cannot find the certificate and private key for decryption.
CertUtil: -store command completed successfully.
```

> **Note**
>
> 次の手順で使用するため、証明書のシリアル番号を控えておいてください。

4. 証明書をエクスポートする

   ルート証明書をファイルにエクスポートします。

```
certutil -store Root certificate-serial-number root-certificate-name.cer
```

5. HSM バックエンド証明書を検証する

   個人用証明書ストアで HSM バックエンド証明書を確認します。

```
PS C:\Users\Administrator\Desktop> certutil -store My

my "Personal"
================ Certificate 0 ================
Serial Number: certificate-serial-number
Issuer: CN=MYRootCA
 NotBefore: 2/5/2020 1:38 PM
 NotAfter: 2/5/2021 1:48 PM
Subject: CN=www.mydomain.com, OU=Certificate Management, O=Information Technology,
 L=Houston, S=Texas, C=US
Non-root Certificate
Cert Hash(sha1): cert-hash
  Key Container = key-container-name
  Provider = Cavium Key Storage Provider
Private key is NOT exportable
Encryption test passed
CertUtil: -store command completed successfully.
```

> **Note**
>
> 次の手順で使用するため、証明書のシリアル番号を控えておいてください。

6. HSM バックエンド証明書をエクスポートする

   HSM バックエンド証明書をファイルにエクスポートします。

   ```
   certutil -store My certificate-serial-number signed-certificate-name.cer
   ```

7. ルート証明書をインポートする

   新しい Windows インスタンス上で次を実行します。

   1. ルート CA ファイルを新しい Windows インスタンスにコピーします
   2. 証明書をインポートします。

      ```
      certutil -addstore Root root-certificate-name.cer
      ```

8. ルート証明書のインストールを確認する

   ルート証明書が正しくインストールされていることを確認します。

   ```
   PS C:\Users\Administrator\Desktop> certutil -store Root

   Root "Trusted Root Certification Authorities"
   ================ Certificate 0 ================
   Serial Number: certificate-serial-number
   Issuer: CN=MYRootCA
    NotBefore: 2/5/2020 1:38 PM
    NotAfter: 2/5/2021 1:48 PM
    Issuer: CN=MYRootCA
   Signature matches Public Key
   Root Certificate: Subject matches Issuer
   Cert Hash(sha1): cert-hash
   No key provider information
   Cannot find the certificate and private key for decryption.
   CertUtil: -store command completed successfully.
   ```

9. HSM バックエンド証明書をインポートする

新しい Windows インスタンス上で次を実行します。

1. HSM バックエンド証明書を新しい Windows インスタンスにコピーします
2. 証明書をインポートします。

```
certutil -addstore My signed-certificate-name.cer
```

10. HSM バックエンド証明書のインストールを検証する

HSM バックエンド証明書が正しくインストールされていることを確認します。

```
PS C:\Users\Administrator\Desktop> certutil -store My

my "Personal"
================ Certificate 0 ================
Serial Number: certificate-serial-number
Issuer: CN=MYRootCA
 NotBefore: 2/5/2020 1:38 PM
 NotAfter: 2/5/2021 1:48 PM
Subject: CN=www.mydomain.com, OU=Certificate Management, O=Information Technology,
 L=Houston, S=Texas, C=US
Non-root Certificate
Cert Hash(sha1): cert-hash
No key provider information
Cannot find the certificate and private key for decryption.
CertUtil: -store command completed successfully.
```

> **Note**
>
> 今後の手順で使用するため、証明書のシリアル番号を控えておいてください。

11. キー参照ファイルを作成する (オプション)

この手順は、新しいキー参照ファイルを作成する必要がある場合にのみ実行してください。それ以外の場合は、次のステップに進みます。

> **Note**
>
> この機能は、SDK バージョン 5.16.0 以降でのみ使用できます。

1. OpenSSL をインストールし、modulus を抽出します。

```
openssl x509 -in signed-certificate-name.cer -modulus -noout
```

> **Note**
>
> OpenSSL コマンドは、modulus を次の形式で出力します: Modulus=*modulus-value*。次のコマンドで使用するため、*modulus-value* を控えておいてください。

2. CloudHSM CLI を使用してキー参照ファイルを作成します。KSP キー参照を生成します (Windows) を参照してください。

```
& "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" key generate-
file --encoding ksp-key-reference --filter attr.class=private-key
 attr.modulus=0xmodulus-value
```

> **Note**
>
> CloudHSM CLI のコマンド引数で使用する *modulus-value* には、16 進数形式を示すために先頭に 0x を付ける必要があります。
> キー参照ファイルは C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition で作成されます。

12. 修復用設定を作成する

以下の内容で repair.txt という名前のファイルを作成します。

```
[Properties]
11 = "" ; Add friendly name property
2 = "{text}" ; Add Key Provider Information property
_continue_="Container=key-container-name&"
_continue_="Provider=Cavium Key Storage Provider&"
_continue_="Flags=0&"
_continue_="KeySpec=2"
```

> **Note**
>
> *key-container-name* を C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition のキー参照ファイル名に置き換えます。

13. 証明書ストアを修復する

    修復コマンドを実行します。

```
certutil -repairstore My certificate-serial-number repair.txt
```

> **Note**
>
> 証明書のシリアル番号は、HSM バックエンド証明書のインストールを確認する前の手順で取得します。

14. 証明書の関連付けを確認する

    証明書が正しく関連付けられていることを確認します。

```
PS C:\Users\Administrator\Desktop> certutil -store My

my "Personal"
================ Certificate 0 ================
Serial Number: certificate-serial-number
Issuer: CN=MYRootCA
 NotBefore: 2/5/2020 1:38 PM
 NotAfter: 2/5/2021 1:48 PM
Subject: CN=www.mydomain.com, OU=Certificate Management, O=Information Technology,
 L=Houston, S=Texas, C=US
Non-root Certificate
Cert Hash(sha1): cert-hash
  Key Container = key-container-name
  Provider = Cavium Key Storage Provider
Private key is NOT exportable
ERROR: Could not verify certificate public key against private key
CertUtil: -store command completed successfully.
```

    出力に以下が表示されていることを確認します。

- 正しいキーコンテナ名
- Cavium キーストレージプロバイダー
- `ERROR: Could not verify certificate public key against private key` は既知の問題です。問題: 証明書ストアの検証が失敗する を参照してください。

15. アプリケーションをテストする

    移行を完了する前に、次の作業を行ってください。

    1. 開発環境でアプリケーションをテストします
    2. 破壊的変更に対応するため、コードを更新します
    3. アプリケーション固有のガイダンスについては、サードパーティーアプリケーションを と統合する AWS CloudHSM を参照してください。

移行を確認する

移行手順を完了したら、以下を確認します。

- 証明書が正しい証明書ストアに正しくインストールされている
- キー参照ファイルが正しい場所にある
- 移行された証明書を使用してアプリケーションが暗号化オペレーションを実行できる

トラブルシューティング

移行中に問題が発生した場合は、以下を確認します。

- すべての証明書がソースシステムから適切にエクスポートされている
- 証明書のシリアル番号がシステム間で一致している
- repair.txt ファイル内のキーコンテナ名がキー参照ファイルと一致している
- SDK3 で生成されたキー参照ファイルを使用している場合は、SDK3 互換モードが有効になっている

関連トピック

- のベストプラクティス AWS CloudHSM

JCE プロバイダーを AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行する

このトピックを使用して、JCE プロバイダーを AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 に移行します。移行の利点については、「AWS CloudHSM クライアント SDK 5 の利点」を参照してください。

では AWS CloudHSM、顧客アプリケーションは AWS CloudHSM クライアントソフトウェア開発キット (SDK) を使用して暗号化オペレーションを実行します。クライアント SDK 5 は、新しい機能とプラットフォームサポートが継続的に追加される、主要な SDK です。

クライアント SDK 3 JCE プロバイダーは、標準 JCE 仕様に含まれていないカスタムクラスと API を使用します。JCE プロバイダーのクライアント SDK 5 は JCE 仕様に準拠しており、特定の領域ではクライアント SDK 3 と下位互換性がありません。お客様のアプリケーションでは、クライアント SDK 5 への移行の一環として変更が必要になる場合があります。このセクションでは、移行を成功させるために必要な変更点を概説します。

すべてのプロバイダーの移行手順を確認するには、「」を参照してくださいAWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行。

トピック

- 重大な変更に対処して準備する
- クライアント SDK 5 への移行
- 関連トピック

重大な変更に対処して準備する

これらの重大な変更を確認し、それに応じて開発環境でアプリケーションを更新します。

プロバイダークラスと名前が変更されました

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
プロバイダークラスと名前	クライアント SDK 3 の JCE プロバイダークラスは `CaviumPro`	クライアント SDK 5 では、プロバイダークラスは `CloudHsmP`	`CloudHsmP rovider` オブジェクトを初期化

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
	vider と呼ばれ、プロバイダー名は Cavium です。	rovider と呼ばれ、プロバイダー名 CloudHSM があります。	する方法は、AWS CloudHSM GitHub サンプルリポジトリを参照してください。

明示的なログインが変更されましたが、暗黙的なログインは従来通りです。

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
明示的なログイン	クライアント SDK 3 は、明示的なログイン[1] に LoginManager クラスを使用します。	クライアント SDK 5 では、CloudHSM プロバイダーは明示的なログインのために AuthProvider を実装します。AuthProvider は標準の Java クラスであり、Java の慣用的な方法でプロバイダーにログインします。クライアント SDK 5 でログイン状態管理が改善されたため、アプリケーションは再接続中にログインをモニタリングして実行する必要がなくなりました[2]。	クライアント SDK 5 で明示的なログインを使用する方法の例については、AWS CloudHSM GitHub サンプルリポジトリの LoginRunner サンプルを参考にしてください。
暗黙的なログイン	暗黙的なログインには変更は必要ありません。クライアント SDK 3 からクライアント SDK 5		クライアント SDK 5 で暗黙的なログイン

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
	に移行する場合、暗黙的なログインでも同じプロパティファイルとすべての環境変数が引き続き機能します。		を使用する方法の例については、 AWS CloudHSM GitHub サンプルリポジトリの LoginRunner サンプルを参照してください。

- [1] クライアント SDK 3 コードスニペット:

```
LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);
```

- [2] クライアント SDK 5 コードスニペット:

```
// Construct or get the existing provider object
AuthProvider provider = new CloudHsmProvider();

// Call login method on the CloudHsmProvider object
// Here loginHandler is a CallbackHandler
provider.login(null, loginHandler);
```

クライアント SDK 5 で明示的なログインを使用する方法の例については、 AWS CloudHSM GitHub サンプルリポジトリの LoginRunner サンプルを参照してください。

キー生成が変更されました

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
キー生成	クライアント SDK 3 では、 `Cavium[Key-type]Al`	Client SDK 5 では、キー生成属性の指定に `KeyAttributesMap` が使用さ	`KeyAttributesMap` を使用して対称キーを生成する方法の例について

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
	gorithmPa rameterSpec はキー生成パラメータの指定に使用されます。コードスニペットについては、脚注 1 を参照してください。	れます。コードスニペットについては、脚注 2 を参照してください。	は、AWS CloudHSM Github サンプルリポジトリの Symmetric Keys サンプルを参考にしてください。
キーペアの生成	クライアント SDK 3 では、Cavium[Key-type]Al gorithmpa rameterSpec を使用してキーペア生成パラメータを指定します。コードスニペットについては、脚注 3 を参照してください。	Client SDK 5 では、これらのパラメーターの指定に KeyPairAt tributesMap が使用されます。コードスニペットについては、脚注 4 を参照してください。	KeyAttrib utesMap を使用して非対称キーを生成する方法の例については、AWS CloudHSM GitHub サンプルリポジトリの AsymmetricKeys サンプルを参照してください。

- [1] クライアント SDK 3 キー生成のコードスニペット:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
keySizeInBits,
keyLabel,
isExtractable,
isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] クライアント SDK 5 キー生成のコードスニペット:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES",
```

```
CloudHsmProvider.PROVIDER_NAME);

final KeyAttributesMap aesSpec = new KeyAttributesMap();
aesSpec.put(KeyAttribute.LABEL, keyLabel);
aesSpec.put(KeyAttribute.SIZE, keySizeInBits);
aesSpec.put(KeyAttribute.EXTRACTABLE, isExtractable);
aesSpec.put(KeyAttribute.TOKEN, isPersistent);

keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [3] クライアント SDK 3 キーペア生成のコードスニペット:

```
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
CaviumRSAKeyGenParameterSpec spec = new CaviumRSAKeyGenParameterSpec(
keySizeInBits,
new BigInteger("65537"),
label + ":public",
label + ":private",
isExtractable,
isPersistent);

keyPairGen.initialize(spec);

keyPairGen.generateKeyPair();
```

- [4] クライアント SDK 5 キーペア生成のコードスニペット:

```
KeyPairGenerator keyPairGen =
KeyPairGenerator.getInstance("RSA", providerName);

// Set attributes for RSA public key
final KeyAttributesMap publicKeyAttrsMap = new KeyAttributesMap();
publicKeyAttrsMap.putAll(additionalPublicKeyAttributes);
publicKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Public");
publicKeyAttrsMap.put(KeyAttribute.MODULUS_BITS, keySizeInBits);
publicKeyAttrsMap.put(KeyAttribute.PUBLIC_EXPONENT,
new BigInteger("65537").toByteArray());

// Set attributes for RSA private key
final KeyAttributesMap privateKeyAttrsMap = new KeyAttributesMap();
privateKeyAttrsMap.putAll(additionalPrivateKeyAttributes);
privateKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Private");
```

```
// Create KeyPairAttributesMap and use that to initialize the
// keyPair generator
KeyPairAttributesMap keyPairSpec =
new KeyPairAttributesMapBuilder()
.withPublic(publicKeyAttrsMap)
.withPrivate(privateKeyAttrsMap)
.build();

keyPairGen.initialize(keyPairSpec);
keyPairGen.generateKeyPair();
```

キーの検索、削除、参照が変更されました

で既に生成されたキーを検索するには、KeyStore AWS CloudHSM を使用します。クライアント SDK 3 には、 `Cavium` と `CloudHSM` の 2 つの KeyStore タイプがあります。クライアント SDK 5 には KeyStore タイプが 1 つだけあります: `CloudHSM`。

`Cavium` KeyStore から `CloudHSM` KeyStore に移動するには、キーストアタイプを変更する必要があります。さらに、クライアント SDK 3 ではキーハンドルでキーを参照していたのに対し、クライアント SDK 5 ではキーラベルを使用するようになりました。これにより、以下の動作が変更されています。

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
キーリファレンス	クライアント SDK 3 では、アプリケーションはキーラベルまたはキーハンドルを使用して HSM 内のキーを参照します。KeyStore でラベルを使用してキーを検索するか、ハンドルを使用して `CaviumKey` オブジェクトを作成します。	クライアント SDK 5 では、アプリケーションは クライアント SDK 5 のAWS CloudHSM KeyStore Java クラス を使用してラベルでキーを検索できます。ハンドルでキーを検索するには、 で AWS CloudHSM `KeyStoreWithAttrib`	

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
		utes を使用します AWS CloudHSM KeyReferenceSpec 。	
複数のエントリの検索	同じ基準を持つ複数の項目が Cavium KeyStore に存在するシナリオで getEntry、getKey、または getCertificate を使用してキーを検索すると、最初に見つかったエントリのみが返されます。	および AWS CloudHSM KeyStoreの場合KeyStoreWithAttributes 、この同じシナリオでは例外がスローされます。この問題を解決するには、CloudHSM CLI で CloudHSM CLI でキーの属性を設定する コマンドを使用してキーに一意のラベルを設定することをお勧めします。または、KeyStoreWithAttributes#getKeys を使用して、基準に一致するすべてのキーを返します。	

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
すべてのキーを検索する	クライアント SDK 3 では、 を使用して HSM 内のすべてのキーを検索できますUtil.findAllKeys() 。	クライアント SDK 5 では、 KeyStoreWithAttributes クラスを使用することで、キーの検索がより簡単かつ効率的になります。可能であれば、レイテンシーを最小限に抑えるためにキーをキャッシュします。詳細については、「アプリケーションのキーを効果的に管理する」を参照してください。HSM からすべてのキーを取得する必要がある場合は、空の KeyAttributesMap を指定して KeyStoreWithAttributes#getKeys を使用します。	KeyStoreWithAttributes クラスを使用してキーを検索する例は、AWS CloudHSM Github サンプルリポジトリで入手できます。コードスニペットは 1 に示されています。

変更点	クライアント SDK 3 の内容	クライアント SDK 5 の内容	例
キー削除	クライアント SDK 3 は `Util.deleteKey()` を使用してキーを削除します。	クライアント SDK 5 の Key オブジェクトは、このDestroyable インターフェイスの `destroy()` メソッドを使用してキーを削除できるインターフェイスを実装します。	キーの削除機能を示すサンプル コードは、CloudHSM Github サンプルリポジトリにあります。各 SDK のサンプルスニペットを 2 に示します。

- [1] スニペットを次に示します。

```
KeyAttributesMap findSpec = new KeyAttributesMap();
findSpec.put(KeyAttribute.LABEL, label);
findSpec.put(KeyAttribute.KEY_TYPE, keyType);
KeyStoreWithAttributes keyStore = KeyStoreWithAttributes.getInstance("CloudHSM");

keyStore.load(null, null);
keyStore.getKey(findSpec);
```

- [2] クライアント SDK 3 でキーを削除する:

```
Util.deleteKey(key);
```

  クライアント SDK 5 でキーを削除する:

```
((Destroyable) key).destroy();
```

暗号アンラップオペレーションが変更されましたが、他の暗号オペレーションは変更されていません

> **Note**
>
> Cipher の暗号化/復号化/ラップオペレーションに変更は必要ありません。

アンラップオペレーションでは、クライアント SDK 3 CaviumUnwrapParameterSpec クラスを、リストされている暗号化オペレーションに固有の次のクラスのいずれかに置き換える必要があります。

- AES/GCM/NoPadding ラップ解除のための GCMUnwrapKeySpec
- AESWrap unwrap と AES/CBC/NoPadding unwrap の場合は IvUnwrapKeySpec
- RSA OAEP unwrap 用の OAEPUnwrapKeySpec

OAEPUnwrapkeySpec のス二ペットの例

```
OAEPParameterSpec oaepParameterSpec =
new OAEPParameterSpec(
        "SHA-256",
        "MGF1",
        MGF1ParameterSpec.SHA256,
        PSpecified.DEFAULT);

KeyAttributesMap keyAttributesMap =
        new KeyAttributesMap(KeyAttributePermissiveProfile.KEY_CREATION);
keyAttributesMap.put(KeyAttribute.TOKEN, true);
keyAttributesMap.put(KeyAttribute.EXTRACTABLE, false);

OAEPUnwrapKeySpec spec = new OAEPUnwrapKeySpec(oaepParameterSpec,
        keyAttributesMap);

Cipher hsmCipher =
        Cipher.getInstance(
                "RSA/ECB/OAEPPadding",
                CloudHsmProvider.PROVIDER_NAME);
hsmCipher.init(Cipher.UNWRAP_MODE, key, spec);
```

署名オペレーションは変更されていません

署名オペレーションに変更は必要ありません。

クライアント SDK 5 への移行

このセクションの指示に従って、クライアント SDK 3 から クライアント SDK 5 に移行します。

> **Note**
>
> Amazon Linux、Ubuntu 16.04、Ubuntu 18.04 CentOS 6、CentOS 8、および RHEL 6 は現在、クライアント SDK 5 ではサポートされていません。現在、クライアント SDK 3 でこれらのプラットフォームのいずれかを使用している場合は、クライアント SDK 5 に移行するときに別のプラットフォームを選択する必要があります。

1. クライアント SDK 3 向けの JCE プロバイダーをアンインストールします。

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client-jce
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client-jce
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client-jce
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-client-jce
   ```

   Ubuntu 16.04 LTS

   ```
   $ sudo apt remove cloudhsm-client-jce
   ```

   Ubuntu 18.04 LTS

   ```
   $ sudo apt remove cloudhsm-client-jce
   ```

2. クライアント SDK 3 の Client Daemon を停止します。

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

3. クライアント SDK 3 の Client Daemon をアンインストールします。

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client
```

> **Note**
>
> カスタム設定を再度有効にする必要があります。

4. 「AWS CloudHSM クライアント SDK 5 の JCE プロバイダーをインストールする」の手順に従って、クライアント SDK JCE プロバイダーをインストールします。
5. クライアント SDK 5 では、新しい設定ファイル形式とコマンドラインブートストラップツールが導入されました。クライアント SDK 5 JCE プロバイダーをブートストラップするには、ユーザー ガイドの 「クライアント SDK をブートストラップする」に記載されている手順に従ってください。
6. アプリケーションは開発環境で実行されていること。既存のコードを更新して、最終的な移行前に重大な変更を解決します。

関連トピック

- のベストプラクティス AWS CloudHSM

クライアント SDK 5 を使用して を操作する AWS CloudHSM

AWS CloudHSM には、2 つの主要なクライアント SDK バージョンが含まれています。

- クライアント SDK 5: これは最新かつデフォルトの クライアント SDK です。クライアント SDK 5 による利点については、「AWS CloudHSM クライアント SDK 5 の利点」を参照してください。

- クライアント SDK 3: これは古いクライアント SDK です。プラットフォームおよび言語ベースのアプリケーションの互換性および管理ツール用のコンポーネントの完全なセットが含まれています。

クライアント SDK 3 から クライアント SDK 5 に移行する手順については、「AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行」を参照してください。

このトピックでは、クライアント SDK 5 について説明します。使用しているクライアント SDK のバージョンを確認するには、次を参照してください。

トピック

- AWS CloudHSM クライアント SDK 5 の利点
- AWS CloudHSM クライアント SDK 5 がサポートするプラットフォーム
- AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリ
- AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine
- AWS CloudHSM クライアント SDK 5 の OpenSSL プロバイダー
- AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP)
- AWS CloudHSM クライアント SDK 5 の JCE プロバイダー

AWS CloudHSM クライアント SDK 5 の利点

AWS CloudHSM Client SDK 3 と比較して、Client SDK 5 は管理が容易で、優れた設定可能性と信頼性を提供します。クライアント SDK 5 には、クライアント SDK 3 には他にも主要な利点がいくつかあります。

サーバーレスアーキテクチャ向けの設計

クライアント SDK 5 にはクライアントデーモンが必要ないため、バックグラウンドサービスを管理する必要がなくなりました。これは、ユーザーにとって次に挙げる重要なポイントで役立ちます。

- アプリケーションの起動プロセスを簡素化します。CloudHSM を使い始めるために必要なことは、アプリケーションを実行する前に SDK を設定することだけです。
- プロセスを常に実行する必要がないため、Lambda や Elastic Container Service (ECS) などのサーバーレスコンポーネントとの統合が容易になります。

サードパーティとの統合が強化され、ポータビリティが容易に

クライアント SDK 5 は JCE 仕様に厳密に準拠しており、異なる JCE プロバイダー間のポータビリティが容易で、サードパーティとの統合も良好です。

ユーザーエクスペリエンスと設定可能性が向上

クライアント SDK 5 では、ログメッセージの読みやすさが向上し、例外やエラー処理メカニズムが明確になり、ユーザーによるセルフサービスのトリアージがはるかに簡単になりました。SDK 5 にはさまざまな設定も用意されており、設定ツールページに一覧表示されています。

より広範なプラットフォームサポート

クライアント SDK 5 は、最新のオペレーティングプラットフォームにより幅広いサポートを提供しています。これには、ARM テクノロジーのサポートのほか、JCE、PKCS #11、および OpenSSL のサポートの強化が含まれます。詳細については、「Supported platforms」を参照してください。

IPv6 接続のサポート

クライアント SDK 5.14 以降では、IPv6 を使用したデュアルスタック HSM への接続がサポートされています。

その他の機能やメカニズム

クライアント SDK 5 には、クライアント SDK 3 にはない機能やメカニズムが追加されており、クライアント SDK 5 には今後もメカニズムが追加されていく予定です。

AWS CloudHSM クライアント SDK 5 がサポートするプラットフォーム

基本サポートは、 AWS CloudHSM クライアント SDK のバージョンごとに異なります。SDK 内のコンポーネントのプラットフォームのサポートは通常、基本サポートと一致しますが、必ずしもそうとは限りません。特定のコンポーネントのプラットフォームのサポートを確認するには、まず目的のプラットフォームが SDK のベースセクションに表示されていることを確認し、コンポーネントセクションで除外項目やその他の関連情報がないか確認します。

AWS CloudHSM は 64 ビットオペレーティングシステムのみをサポートします。

プラットフォームのサポートは時間の経過とともに変化します。以前のバージョンの CloudHSM クライアント SDK では、ここに記載されているすべてのオペレーティングシステムがサポートされていない場合があります。リリースノートを使用して、以前のバージョンの CloudHSM クライアント SDK に対するオペレーティングシステムサポートを確認します。詳細については、「AWS CloudHSM クライアント SDK のダウンロード」を参照してください。

以前の クライアント SDK でサポートされるプラットフォームについては、「AWS CloudHSM クライアント SDK 3 でサポートされているプラットフォーム」を参照してください。

クライアント SDK 5 にはクライアントデーモンは必要ありません。

トピック

- AWS CloudHSM クライアント SDK 5 の Linux サポート
- AWS CloudHSM クライアント SDK 5 の Windows サポート
- AWS CloudHSM クライアント SDK 5 のサーバーレスサポート
- AWS CloudHSM クライアント SDK 5 の HSM 互換性

AWS CloudHSM クライアント SDK 5 の Linux サポート

AWS CloudHSM クライアント SDK 5 では、次の Linux オペレーティングシステムとプラットフォームがサポートされています。

サポートされているプラットフォーム	x86_64 アーキテクチャ	ARM アーキテクチャ
Amazon Linux 2	はい	はい
Amazon Linux 2023	はい	はい
Red Hat Enterprise Linux 8 (8.3 以降)	はい	はい
Red Hat Enterprise Linux 9 (9.2 以降)	はい	はい
Red Hat Enterprise Linux 10 (10.0 以降)	はい	はい

サポートされているプラットフォーム	x86_64 アーキテクチャ	ARM アーキテクチャ
Ubuntu 22.04 LTS	はい	はい
Ubuntu 24.04 LTS	はい	はい
Ubuntu 26.04 LTS	はい	はい

- SDK 5.16 は、Ubuntu 20.04 LTS プラットフォームをサポートする最後のリリースでした。詳細については、「Ubuntu のウェブサイト」を参照してください。
- SDK 5.12 は、CentOS 7 (7.8 以降) プラットフォームをサポートする最後のリリースでした。詳細については、「CentOS のウェブサイト」を参照してください。
- SDK 5.12 は、Red Hat Enterprise Linux 7 (7.8 以降) プラットフォームをサポートする最後のリリースでした。詳細については、Red Hat のウェブサイトを参照してください。
- SDK 5.4.2 は、CentOS 8 プラットフォームをサポートする最後のリリースでした。詳細については、「CentOS のウェブサイト」を参照してください。

AWS CloudHSM クライアント SDK 5 の Windows サポート

AWS CloudHSM クライアント SDK 5 では、次のバージョンの Windows Server がサポートされています。

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019
- Microsoft Windows Server 2022
- Microsoft Windows Server 2025

AWS CloudHSM クライアント SDK 5 のサーバーレスサポート

AWS CloudHSM クライアント SDK 5 は、次の AWS サーバーレスサービスをサポートしています。

- AWS Lambda
- Docker および ECS

AWS CloudHSM クライアント SDK 5 の HSM 互換性

次の表に、HSMs の AWS CloudHSM クライアント SDK 5 の互換性を示します。

hsm1.medium	hsm2m.medium
クライアント SDK バージョン 5.0.0 以降と互換性があります。	Client SDK バージョン 5.9.0 以降と互換性があります。

AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリ

PKCS #11 は、ハードウェアセキュリティモジュール (HSM) で暗号化オペレーションを実行するための標準です。 AWS CloudHSM は、PKCS #11 バージョン 2.40 に準拠した PKCS #11 ライブラリの実装を提供します。

ブートストラップの詳細については、「クラスターへの接続」を参照してください。トラブルシューティングについては、「の PKCS #11 ライブラリの既知の問題 AWS CloudHSM」を参照してください。

クライアント SDK 3 の使用の詳細については、「以前の SDK バージョンを使用して を操作する AWS CloudHSM」を参照してください。

トピック

- AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリをインストールする
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリへの認証
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされているキータイプ
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされているメカニズム
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされている API オペレーション
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリの主な属性
- AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリのコードサンプル
- の PKCS #11 ライブラリの詳細設定 AWS CloudHSM
- PKCS #11 ライブラリを使用した証明書ストレージ

AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリをインストールする

このトピックでは、 AWS CloudHSM クライアント SDK 5 バージョンシリーズの最新バージョンの PKCS #11 ライブラリをインストールする方法について説明します。クライアント SDK または PKCS #11 ライブラリの詳細については、[クライアント SDK の使用] と [PKCS #11 ライブラリ] を参照してください。

クライアント SDK 5 では、クライアントデーモンをインストールまたは実行する必要はありません。

クライアント SDK 5 で単一の HSM クラスターを実行するには、まず `disable_key_availability_check` を `True` に設定してクライアントキーの耐久性の設定を管理する必要があります。詳細については、キーの同期 と クライアント SDK 5 設定ツール を参照してください。

クライアント SDK 5 の PKCS #11 ライブラリの詳細については、[PKCS #11 ライブラリ] を参照してください。

> **Note**
>
> クライアント SDK 5 で単一の HSM クラスターを実行するには、まず `disable_key_availability_check` を `True` に設定してクライアントキーの耐久性の設定を管理する必要があります。詳細については、キーの同期 と クライアント SDK 5 設定ツール を参照してください。

PKCS #11 ライブラリをインストールおよび設定するには

1. 次のコマンドを使用して、PKCS #11 ライブラリのダウンロードとインストールを行います。

   Amazon Linux 2023

   X86_64 アーキテクチャ上で Amazon Linux 2023 用の PKCS #11 ライブラリをインストールします。

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
   cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
   ```

ARM64 アーキテクチャ上で Amazon Linux 2023 用の PKCS #11 ライブラリをインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

X86_64 アーキテクチャ上で Amazon Linux 2 用の PKCS #11 ライブラリをインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

ARM64 アーキテクチャ上で Amazon Linux 2 用の PKCS #11 ライブラリをインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
pkcs11-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

RHEL 10 用の PKCS #11 ライブラリを X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/
cloudhsm-pkcs11-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el10.x86_64.rpm
```

RHEL 10 用の PKCS #11 ライブラリを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/
cloudhsm-pkcs11-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

RHEL 9 用の PKCS #11 ライブラリーを X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-
pkcs11-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.x86_64.rpm
```

RHEL 9 用の PKCS #11 ライブラリーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-
pkcs11-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

RHEL 8 用の PKCS #11 ライブラリーを X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-
pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el8.x86_64.rpm
```

RHEL 8 用の PKCS #11 ライブラリを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-
pkcs11-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el8.aarch64.rpm
```

Ubuntu 26.04 LTS

Ubuntu 26.04 LTS 用の PKCS #11 ライブラリを X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/
cloudhsm-pkcs11_latest_u26.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u26.04_amd64.deb
```

Ubuntu 26.04 LTS 用の PKCS #11 ライブラリを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/
cloudhsm-pkcs11_latest_u26.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u26.04_arm64.deb
```

Ubuntu 24.04 LTS

Ubuntu 24.04 LTS 用の PKCS #11 ライブラリを X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/
cloudhsm-pkcs11_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u24.04_amd64.deb
```

Ubuntu 24.04 LTS 用の PKCS #11 ライブラリを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/
cloudhsm-pkcs11_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS 用の PKCS #11 ライブラリを X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

Ubuntu 22.04 LTS 用の PKCS #11 ライブラリを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

Windows Server

Windows Server 用の PKCS #11 ライブラリを X86_64 アーキテクチャにインストールします。

1. [クライアント SDK 5 用の PKCS #11 ライブラリ] をダウンロードします。
2. Windows の管理権限を持つ PKCS #11 ライブラリインストーラ (AWSCloudHSMPKCS11-latest.msi) を実行します。

2. 構成ツールを使用して、証明書の発行場所を指定します。手順については、「発行証明書の場所を指定する」を参照してください。
3. クラスターに接続して使用するには、「クライアント SDK をブートストラップする」を参照してください。
4. PKCS #11 ライブラリのファイルは、次の場所にあります。

   - Linuxのバイナリ、設定スクリプト、およびログファイル:

```
/opt/cloudhsm
```

Windows のバイナリ:

```
C:\Program Files\Amazon\CloudHSM
```

Windows の設定スクリプトとログファイル:

```
C:\ProgramData\Amazon\CloudHSM
```

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリへの認証

PKCS #11 ライブラリを使用すると、アプリケーションは AWS CloudHSMで特定の Crypto User (CU) として実行されます。アプリケーションは、CU が所有して共有するキーのみを表示および管理できます。既存の CU を HSM で使用することも、アプリケーションに新しい CU を作成することもできます。CU の管理については、「CloudHSM CLI による HSM ユーザーの管理」および「CloudHSM 管理ユーティリティ (CMU) による HSM ユーザーの管理」を参照してください

PKCS #11 に CU を指定するには、PKCS #11 [C_Login 関数] のピンパラメーターを使用します。の場合 AWS CloudHSM、ピンパラメータの形式は次のとおりです。

```
<CU_user_name>:<password>
```

たとえば、次のコマンドでユーザー名 CryptoUser とパスワード CUPassword123! を使用して PKCS #11 ライブラリのピンを CU に設定します。

```
CryptoUser:CUPassword123!
```

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされているキータイプ

AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリは、次のキータイプをサポートしています。

キータイプ	説明
AES	128、192、256 ビットの AES キーを生成します。
Triple DES (3DES、DESede)	192 ビットのトリプル DES キーを生成します。今後の変更については、以下の注記「1」を参照してください。
EC	secp224r1 (P-224)、secp256r1 (P-256)、secp256k1 (ブロックチェーン)、secp384r1 (P-384)、secp521r1 (P-521)、および ed25519[2] 曲線を使用してキーを生成します。
ML-DSA	パラメータセット ML-DSA[3]-ML-DSA-44ML-DSA-65ML-DSA-87キーペアを生成します。
[GENERIC_SECRET]	1 ~ 800 バイトの汎用シークレットを生成します。
RSA	256 ビットの増分で、2,048 ~ 4,096 ビットの RSA キーを生成します。

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

[2] 非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

[3] 2026 年 9 月 1 日以降、ML-DSA は hsm2m.medium クラスターの FIPS モードで使用できます。

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされているメカニズム

PKCS #11 ライブラリは PKCS #11 仕様のバージョン 2.40 に準拠しています。PKCS#11 を使用して暗号化機能を呼び出すには、指定されたメカニズムで関数を呼び出します。以下のセクションでは、 AWS CloudHSM クライアント SDK 5 でサポートされている関数とメカニズムの組み合わせをまとめます。

PKCS #11 ライブラリは、次のアルゴリズムをサポートしています。

- [暗号化と復号化] AES-CBC、AES-CTR、AES-ECB、AES-GCM、DES3-CBC、DES3-ECB、RSA-OAEP、RSA-PKCS
- 署名と検証 – RSA、HMAC、ECDSSA (ハッシュあり/なし)、EdDSA (Ed25519 および Ed25519ph)、ML-DSA (純粋および外部 mu モード)
- [ハッシュ/ダイジェスト] SHA1、SHA224、SHA256、SHA384、SHA512
- [キーラップ] AES キーラップ、[1] AES-GCM、RSA-AES、RSA-OAEP
- キー導出 — SP800-108 Counter KDF、および KDF 付き ECDH (サポートされている KDF アルゴリズムは、X9.63 with SHA1、SHA224、SHA256、SHA384、SHA512 です)

トピック

- キーとキーペアの関数を生成する
- 署名および検証
- リカバリ機能への署名、リカバリ、検証
- ダイジェスト関数
- 暗号化と復号
- キー機能の導出
- ラップ関数とアンラップ関数
- 各メカニズムの最大データサイズ
- メカニズムの注釈

キーとキーペアの関数を生成する

PKCS #11 ライブラリ用の AWS CloudHSM ソフトウェアライブラリを使用すると、キーとキーペアの生成関数に次のメカニズムを使用できます。

- `CKM_RSA_PKCS_KEY_PAIR_GEN`
- `CKM_RSA_X9_31_KEY_PAIR_GEN` – このメカニズムは機能的には `CKM_RSA_PKCS_KEY_PAIR_GEN` メカニズムと似ていますが、p と q の生成に関してより強力な保証を提供します。
- `CKM_EC_KEY_PAIR_GEN`
- `CKM_CLOUDHSM_ML_DSA_KEY_PAIR_GEN`[3]

- CKM_GENERIC_SECRET_KEY_GEN
- CKM_AES_KEY_GEN
- CKM_DES3_KEY_GEN— 今後の変更は脚注 5 に記載されています。

署名および検証

PKCS #11 ライブラリ用の AWS CloudHSM ソフトウェアライブラリを使用すると、署名および検証関数に次のメカニズムを使用できます。クライアント SDK 5 では、データはソフトウェアでローカルにハッシュされます。つまり、SDK でハッシュできるデータのサイズに制限はありません。

クライアント SDK 5 では、RSA と ECDSA のハッシュはローカルで行われるため、データ制限はありません。HMAC と PureEdDSA (Ed25519) では、データ制限があります。詳細については、脚注 2 を参照してください。

RSA

- CKM_RSA_X_509
- CKM_RSA_PKCS – シングルパートのオペレーションのみ。
- CKM_RSA_PKCS_PSS – シングルパートのオペレーションのみ。
- CKM_SHA1_RSA_PKCS
- CKM_SHA224_RSA_PKCS
- CKM_SHA256_RSA_PKCS
- CKM_SHA384_RSA_PKCS
- CKM_SHA512_RSA_PKCS
- CKM_SHA512_RSA_PKCS
- CKM_SHA1_RSA_PKCS_PSS
- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS

ECDSA

- CKM_ECDSA – シングルパートのオペレーションのみ。

- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512

ML-DSA

- CKM_CLOUDHSM_ML_DSA_KEY_PAIR_GEN[3] – ML-DSA キーペアを生成します。
- CKM_CLOUDHSM_ML_DSA[3] – 純粋な ML-DSA 署名と検証。
- CKM_CLOUDHSM_EXT_MU_ML_DSA[3] – ML-DSA 署名と外部 mu による検証。

EdDSA

- CKM_CLOUDHSM_EDDSA[6][7] — NULL パラメータ (PureEdDSA) の場合、単一パートオペレーションのみ。

HMAC

- CKM_SHA_1_HMAC[2]
- CKM_SHA224_HMAC[2]
- CKM_SHA256_HMAC[2]
- CKM_SHA384_HMAC[2]
- CKM_SHA512_HMAC[2]

CMAC

- CKM_AES_CMAC

リカバリ機能への署名、リカバリ、検証

クライアント SDK 5 は、署名と復号機能をサポートしていません。

ダイジェスト関数

PKCS #11 ライブラリ用の AWS CloudHSM ソフトウェアライブラリを使用すると、ダイジェスト関数に次のメカニズムを使用できます。クライアント SDK 5 では、データはソフトウェアでローカルにハッシュされます。つまり、SDK でハッシュできるデータのサイズに制限はありません。

- CKM_SHA_1
- CKM_SHA224
- CKM_SHA256
- CKM_SHA384
- CKM_SHA512

暗号化と復号

PKCS #11 ライブラリ用の AWS CloudHSM ソフトウェアライブラリを使用すると、暗号化および復号関数に次のメカニズムを使用できます。

- CKM_RSA_X_509
- CKM_RSA_PKCS – シングルパートのオペレーションのみ。今後の変更は脚注 5 に記載されています。
- CKM_RSA_PKCS_OAEP – シングルパートのオペレーションのみ。
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC— 今後の変更は脚注 5 に記載されています。
- CKM_DES3_ECB— 今後の変更は脚注 5 に記載されています。
- CKM_DES3_CBC_PAD— 今後の変更は脚注 5 に記載されています。
- CKM_AES_GCM [1], [2]
- CKM_CLOUDHSM_AES_GCM[3]

キー機能の導出

PKCS #11 ライブラリ用の AWS CloudHSM ソフトウェアライブラリは、次のキー取得メカニズムをサポートしています。

- CKM_SP800_108_COUNTER_KDF
- CKM_ECDH1_DERIVE - 次のベンダー定義の KDF タイプを使用した ECDH キー導出をサポートします[6]。
  - CKD_CLOUDHSM_X963_SHA1_KDF - X9.63 KDF with SHA1[8]
  - CKD_CLOUDHSM_X963_SHA224_KDF - X9.63 KDF with SHA224[8]
  - CKD_CLOUDHSM_X963_SHA256_KDF - X9.63 KDF with SHA256[8]
  - CKD_CLOUDHSM_X963_SHA384_KDF - X9.63 KDF with SHA384[8]
  - CKD_CLOUDHSM_X963_SHA512_KDF - X9.63 KDF with SHA512[8]

ラップ関数とアンラップ関数

PKCS #11 ライブラリ用の AWS CloudHSM ソフトウェアライブラリを使用すると、ラップ関数とラップ解除関数に次のメカニズムを使用できます。

AES キーラップに関する追加情報については、[AES キーラップ] を参照してください。

- CKM_RSA_PKCS – シングルパートのオペレーションのみ。今後の変更は、脚注 5 に記載されています。
- CKM_RSA_PKCS_OAEP[4]
- CKM_AES_GCM[1, 3]
- CKM_CLOUDHSM_AES_GCM[3]
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD[3]
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD[3]
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD[3]

各メカニズムの最大データサイズ

次の表は、各メカニズムに設定されている最大データサイズを示します:

最大データセットサイズ

メカニズム	最大データサイズ (バイト単位)
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280
CKM_CLOUDHSM_EDDSA (PureEdDSA、NULL パラメータ)	16000

メカニズムの注釈

- [1] AES-GCM の暗号化を実行している際、HSM はアプリケーションからの初期化ベクトル (IV) データを受け入れません。HSM が生成した IV を使用する必要があります。HSM で生成された 12 バイトの IV は、指定した CK_GCM_PARAMS パラメータ構造の pIV 要素が指すメモリ参照に書き込まれます。ユーザーが混乱しないよう、バージョン 1.1.1 以降の PKCS#11 SDK では、AES-GCM 暗号化が初期化されると、pIV はゼロ化されたバッファを指し示すようになっています。
- [2] 以下の仕組みでデータをオペレーションする場合、データバッファが最大データサイズを超えるとオペレーションがエラーとなります。これらのメカニズムでは、すべてのデータ処理が HSM 内で行われる必要があります。各メカニズムの最大データサイズセットについては、「各メカニズムの最大データサイズ」を参照してください。

- [3] ベンダー定義のメカニズム。CloudHSM ベンダー定義のメカニズムを使用するには、コンパイル時に PKCS #11 アプリケーションに /opt/cloudhsm/include/pkcs11/pkcs11t.h を含める必要があります。

  CKM_CLOUDHSM_AES_GCM: この独自のメカニズムは、標準 CKM_AES_GCM よりもプログラム的に安全な代替手段です。これは、HSM によって生成された IV を、暗号の初期化中に提供される CK_GCM_PARAMS 構造体に書き戻すのではなく、暗号文の先頭に付加します。このメカニズムは C_Encrypt、C_WrapKey、C_Decrypt、C_UnwrapKey 関数で使用できます。このメカニズムを使用する場合は、CK_GCM_PARAMS 構造体内の pIV 変数を NULL に設定する必要があります。このメカニズムを C_Decrypt および C_UnwrapKey と共に使用する場合、IV は、ラップ解除される暗号文の前に付加されることが想定されます。

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: PKCS #5 パディングを使用する AES キーラップ。

  CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: ゼロパディングを使用する AES キーラップ。

  CKM_CLOUDHSM_ML_DSA_KEY_PAIR_GEN: ML-DSA キーペア
  を生成します。CKA_CLOUDHSM_PARAMETER_SET 属性 (タイプ
  CK_CLOUDHSM_ML_DSA_PARAMETER_SET_TYPE) を使用して、パラメータセットを指定します。有効な値
  は、CKP_CLOUDHSM_ML_DSA_44、CKP_CLOUDHSM_ML_DSA_65、CKP_CLOUDHSM_ML_DSA_87 です。

  CKM_CLOUDHSM_ML_DSA: 純粋な ML-DSA 署名と検証。ML-DSA キーペアを使用して raw データに署名または検証します。

  CKM_CLOUDHSM_EXT_MU_ML_DSA: ML-DSA 署名と外部 mu による検証。外部 mu 値に事前処理されたデータに署名または検証します。

  CKM_CLOUDHSM_EDDSA: EdDSA 署名メカニズム。NULL パラメータの場合: PureEdDSA (Ed25519)、シングルパートのみ、最大 16,000 バイト。CK_EDDSA_PARAMS (phFlag = CK_TRUE、ulContextDataLen = 0): Ed25519ph、シングルパートとマルチパート、サイズ制限なし。CK_EDDSA_PARAMS 構造は PKCS #11 v3.0 仕様に従います。

- [4] 次の CK_MECHANISM_TYPE および CK_RSA_PKCS_MGF_TYPE
  は、CK_RSA_PKCS_OAEP_PARAMS の CKM_RSA_PKCS_OAEP としてサポートされています:
  - CKG_MGF1_SHA1 を使用する CKM_SHA_1
  - CKG_MGF1_SHA224 を使用する CKM_SHA224

  - CKG_MGF1_SHA256 を使用する CKM_SHA256
  - CKM_MGF1_SHA384 を使用する CKM_SHA384
  - CKM_MGF1_SHA512 を使用する CKM_SHA512
- [5] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。
- [6] ベンダー定義タイプ。CloudHSM ベンダー定義タイプを使用するには、コンパイル時に PKCS #11 アプリケーションに cloudhsm_pkcs11_vendor_defs.h を含める必要があります。これは、Linux ベースのプラットフォームでは /opt/cloudhsm/include/pkcs11/cloudhsm_pkcs11_vendor_defs.h に、Windows ベースのプラットフォームでは C:\Program Files\Amazon\CloudHSM\include\pkcs11\cloudhsm_pkcs11_vendor_defs.h にあります。
- [7] 非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。
- [8] キー取得関数 (KDFs) は NIST Special Publication 800-56A Revision 3 で指定されています。

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされている API オペレーション

PKCS #11 ライブラリは、 AWS CloudHSM クライアント SDK 5 の次の PKCS #11 API オペレーションをサポートしています。

- C_CloseAllSessions
- C_CloseSession
- C_CreateObject
- C_Decrypt
- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DeriveKey
- C_DestroyObject
- C_Digest
- C_DigestFinal

- C_DigestInit
- C_DigestUpdate
- C_Encrypt
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate
- C_Finalize
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal

- C_SignInit
- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyUpdate
- C_WrapKey

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリの主な属性

AWS CloudHSM キーオブジェクトは、パブリックキー、プライベートキー、シークレットキーのいずれかです。キーオブジェクトで許可されているアクションは属性で指定されます。属性は、キーオブジェクトの作成時に定義されます。PKCS #11 ライブラリを に使用すると AWS CloudHSM、PKCS #11 標準で指定されたデフォルト値が割り当てられます。

AWS CloudHSM は、PKCS #11 仕様に記載されているすべての属性をサポートしているわけではありません。サポートするすべての属性の仕様に準拠しています。これらの属性は、それぞれのテーブルにリストされています。

オブジェクトを作成、変更、またはコピーする C_CreateObject、C_GenerateKey、C_GenerateKeyPair、C_UnwrapKey、C_DeriveKey などの暗号化関数は、属性テンプレートをパラメータの 1 つとして使用します。オブジェクトの作成中に属性テンプレートを渡す方法の詳細については、「PKCS #11 ライブラリを使用したキーの生成」のサンプルを参照してください。

以下のトピックでは、 AWS CloudHSM キー属性の詳細と、キーオブジェクトと証明書オブジェクトの属性の読み取りについて説明します。

トピック

- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリを使用して属性を取得する
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリ属性テーブル
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリ属性の変更
- AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリエラーコードの解釈

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリを使用して属性を取得する

アプリケーションがキーまたは証明書オブジェクトから属性を読み取ると、PKCS #11 ライブラリはC_GetAttributeValue呼び出しごとに 1 つのリクエストをクラスターに送信します。テンプレート内の属性の数は、リクエストの数を変更しません。10 個の属性をリクエストする 1 つの呼び出しは、1 つのリクエストを送信します。各リクエストで 1 つの属性が 10 個のリクエストを送信する 10 個の呼び出し。

各呼び出しは、1 つのリクエストでクラスターからオブジェクトの属性を読み取り、テンプレートがリクエストする値をコピーします。PKCS #11 ライブラリは値をキャッシュしないため、すべての呼び出しが新しいリクエストを送信します。

アプリケーションが送信するリクエストの数を減らすには、次の推奨事項に従ってください。

- 1 つのテンプレートで必要なすべての属性をリクエストします。必要なすべての属性を含む単一のテンプレートを構築し、1 回のC_GetAttributeValue呼び出しに渡します。次の例では、1 回の呼び出しでキーの 3 つの属性を読み取ります。

```
CK_OBJECT_CLASS obj_class;
CK_BYTE label[128];
CK_BYTE id[128];

CK_ATTRIBUTE attrs[] = {
        {CKA_CLASS, &obj_class, sizeof(obj_class)},
        {CKA_LABEL, label,       sizeof(label)},
        {CKA_ID,    id,          sizeof(id)},
};

rv = funcs->C_GetAttributeValue(session, object, attrs,
                                sizeof(attrs) / sizeof(CK_ATTRIBUTE));
```

  詳細については、GitHub ウェブサイトの「キー属性を一覧表示する」を参照してください。
- オブジェクトがサポートし、読み取れる属性のみをリクエストします。テンプレートがオブジェクトにない属性に名前を付けると、 はその属性CK_UNAVAILABLE_INFORMATIONの ulValueLenを C_GetAttributeValueに設定し、を返しますCKR_ATTRIBUTE_TYPE_INVALID。機密属性は同じように動作し、 を返しますCKR_ATTRIBUTE_SENSITIVE。どちらの場合も、テンプレート内の他の属性は引き続き値を受け取ります。呼び出しの戻り値のみではなく、各属性ulValueLenを確認します。各オブジェクトタイプがサポートする属性と、機密性の高い属性については、「」を参照してくださいAWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリ属性テーブル。

- 言語バインディングが行う呼び出しの数を確認します。Python や Java などの言語の PKCS #11 バインディングは、送信するC_GetAttributeValue呼び出しの数を決定します。一部のバインディングは属性ごとに 1 つの呼び出しを送信し、一部のバインディングは 2 つの呼び出しを送信します。複数の属性を一度に読み取るためにバインディングが提供するインターフェイスを使用し、アプリケーションが送信する呼び出しを測定します。
- アプリケーションで属性値をキャッシュします。オブジェクトの属性を 1 回読み、アプリケーションに必要な値を保存します。キャッシュされた値は、オブジェクトが変更されるまで再利用できます。他のアプリケーションまたはユーザーがオブジェクトを変更できる場合は、変更後にキャッシュを更新します。
- 属性を読み取る前にバッファを割り当てます。PKCS #11 標準は、可変長の値の 2 パスパターンを記述します。アプリケーションは を pValueに設定C_GetAttributeValueして NULL_PTRを呼び出して長さを学習し、再度呼び出して値を読み込みます。どちらの呼び出しもリクエストを送信します。予想される値を保持するバッファサイズがわかっている場合は、まずバッファを割り当て、1 回の呼び出しで値を読み取ります。

  バッファが小さすぎると、その属性CKR_BUFFER_TOO_SMALLに対して が返され、呼び出しは引き続きリクエストを送信します。予想される値に対して十分な余裕のあるバッファサイズを選択します。などのテンプレートを保持する属性にはCKA_WRAP_TEMPLATE、引き続き 2 パスパターンが必要です。

> **Note**
>
> これらの推奨事項は、キーオブジェクトと証明書オブジェクトの両方に適用され、それぞれに異なる制限が適用されます。証明書ストレージには固定読み取りレート制限があり、証明書オブジェクトのC_GetAttributeValue呼び出しごとに 1 回の読み取りオペレーションが使用されます。キーオブジェクトの場合、HSMsリング属性は、固定レートではなく容量に達すると読み取ります。PKCS #11 ライブラリは、これらのリクエストを自動的に再試行します。送信するリクエストの数を減らすと、アプリケーションが両方の制限内に収まるようになります。詳細については、「証明書ストレージの制限」および「HSM スロットリング」を参照してください。

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリ属性テーブル

の PKCS #11 ライブラリテーブルには、キータイプによって異なる属性のリスト AWS CloudHSM が含まれています。これは、特定の暗号化関数を で使用するときに、特定のキータイプで特定の属性がサポートされるかどうかを示します AWS CloudHSM。

凡例:

- ✓ CloudHSM が特定のキータイプの属性をサポートしていることを示します。
- ✖ CloudHSM が特定のキータイプの属性をサポートしていないこと示します。
- R は、属性値が特定のキータイプに対して読み取り専用に設定されていることを示します。
- S は、属性が機密であるため、`GetAttributeValue` で読み取れないことを示します。
- [Default Value] 列のセルが空の場合は、属性に割り当てられている特定のデフォルト値がないことを示します。

GenerateKeyPair

属性	キータイプ						デフォルト値
	EC プライベート	EC パブリック	RSA プライベート	RSA パブリック	ML-DSA プライベート	ML-DSA パブリック	
CKA_CLASS	✓	✓	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	正
CKA_TOKEN	✓	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True

属性	キータイプ						デフォルト値
CKA_ENCRYPT	✖	✓[3]	✖	✓	✖	✖	誤
CKA_DECRYPT	✓[3]	✖	✓	✖	✖	✖	誤
CKA_DERIVE	✓	✓	✓	✓	✖	✖	誤
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	✓	✖	誤
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	✖	
CKA_VERIFY	✖	✓	✖	✓	✖	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✓[3]	✖	✓	✖	✖	誤
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✖	✖	
CKA_TRUSTED	✖	✓	✖	✓	✖	✓	誤

属性	キータイプ						デフォルト値
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✖	誤
CKA_UNWRAP	✓[3]	✖	✓	✖	✖	✖	誤
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✖	✖	
CKA_SENSITIVE	✓[1]	✖	✓[1]	✖	✓	✖	正
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	✖	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✖	正
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	✖	✖	
CKA_MODULUS	✖	✖	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	✖	✖	

属性	キータイプ						デフォルト値
CKA_COEFFICIENT	✖	✖	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	✖	✖	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	✖	✖	
CKA_CLOUDHSM_PARAMETER_SET	✖	✖	✖	✖	✓	✓	

GenerateKey

属性	キータイプ			デフォルト値
	AES	DES3	汎用シークレット	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	正
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	誤
CKA_DECRYPT	✓	✓	✖	誤
CKA_DERIVE	✓	✓	✓	誤
CKA_MODIFIABLE	✓	✓	✓	True
CKA_DESTROYABLE	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	True

属性	キータイプ			デフォルト値
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	True
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	誤
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	誤
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	誤
CKA_UNWRAP	✓	✓	✖	誤
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	✖	✖	✖	

属性	キータイプ			デフォルト値
CKA_EXTRACTABLE	✓	✓	✓	正
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	

属性	キータイプ				デフォルト値
CKA_EC_PARAMS		✖	✖	✖	
CKA_EC_POINT		✖	✖	✖	
CKA_VALUE		✖	✖	✖	
CKA_VALUE_LEN		✓[2]	✖	✓[2]	
CKA_CHECK_VALUE		R	R	R	

CreateObject

> **ML-DSA と CreateObject**
>
> を介した ML-DSA キーのインポートC_CreateObjectはサポートされていません。ML-DSA キーを使用するには、 を使用して HSM で直接生成しますC_GenerateKeyPair。

属性	キータイプ							デフォルト値
	EC プライベート	EC パブリック	RSA プライベート	RSA パブリック	AES	DES3	汎用シークレット	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	

属性	キータイプ							デフォルト値
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	誤
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	誤
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖	誤
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖	誤
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	誤
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	誤
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	✖	✖	誤

属性	キータイプ							デフォルト値
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	誤
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	誤
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	誤
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	誤
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	正
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	正

属性	キータイプ							デフォルト値
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_EC_PARAMS	✓[2]	✓[2]	✖	✖	✖	✖	✖	

属性	キータイプ							デフォルト値
CKA_EC_POINT	✖	✓[2]	✖	✖	✖	✖	✖	
CKA_VALUE	✓[2]	✖	✖	✖	✓[2]	✓[2]	✓[2]	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

属性	キータイプ					デフォルト値
	EC プライベート	RSA プライベート	AES	DES3	汎用シークレット	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	誤

属性	キータイプ					デフォルト値
CKA_TOKEN	✓	✓	✓	✓	✓	誤
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✓	✓	✖	誤
CKA_DECRYPT	✖	✓	✓	✓	✖	誤
CKA_DERIVE	✓	✓	✓	✓	✓	誤
CKA_MODIFIABLE	✓	✓	✓	✓	✓	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	✓	✓	誤
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	誤
CKA_VERIFY	✖	✖	✓	✓	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	誤
CKA_UNWRAP	✖	✓	✓	✓	✖	誤

属性	キータイプ					デフォルト値
CKA_SENSITIVE	✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	正
CKA_NEVER_EXTRACTABLE	R	R	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	
CKA_MODULUS	✖	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	✖	

属性	キータイプ					デフォルト値
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	

DeriveKey

属性	キータイプ				デフォルト値
	AES	DES3	汎用シークレット		
CKA_CLASS	✓[2]	✓[2]	✓[2]		
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]		

属性	キータイプ			デフォルト値
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	正
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	誤
CKA_DECRYPT	✓	✓	✖	誤
CKA_DERIVE	✓	✓	✓	誤
CKA_MODIFIABLE	✓	✓	✓	True
CKA_DESTROYABLE	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	誤
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	

属性	キータイプ			デフォルト値
CKA_WRAP	✓	✓	✖	誤
CKA_UNWRAP	✓	✓	✖	誤
CKA_SENSITIVE	R	R	R	正
CKA_EXTRACTABLE	✓	✓	✓	正
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	

属性	キータイプ			デフォルト値
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

属性	キータイプ								
	EC プライベート	EC パブリック	RSA プライベート	RSA パブリック	ML-DSA プライベート	ML-DSA パブリック	AES	DES3	汎用シークレット

属性	キータイプ								
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	R	R	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓	✓	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✖	✖	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✖	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✖	✖	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✖	✓	✓	✓

属性	キータイプ								
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✖	✖	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✖	✖	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✖	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✖	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✖	✓	✓	✓
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✖	✓	✓	✓

属性	キータイプ								
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	R	R	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖	✖	✖
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖	✖	✖

属性	キータイプ								
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✖	✖	✓	✓	✓
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✖	✖	✓	✓	✖
CKA_CLOUDHSM_PARAMETER_SET	✖	✖	✖	✖	R	R	✖	✖	✖

属性注釈

- [1] この属性はファームウェアによって部分的にサポートされており、デフォルト値にのみ明示的に設定する必要があります。
- [2] 必須属性。
- [3] Ed25519 キー (ed25519 曲線を持つ EC キー): CKA_ENCRYPT、CKA_DECRYPT、CKA_WRAP、CKA_UNWRAP、および CKA_DERIVE は機能しません。Ed25519 キーは署名専用であり、暗号化、ラッピング、またはキー取得オペレーションをサポートしていません。

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリ属性の変更

AWS CloudHSM オブジェクトの PKCS #11 ライブラリ属性の中には、オブジェクトの作成後に変更できるものもあれば、できないものもあります。属性を修正するには、CloudHSM CLI の key set-attribute コマンドを使用します。CloudHSM CLI の key list コマンドを使用して属性のリストを導出することもできます。

次のリストで、オブジェクトの作成後に変更が許可荒れている許可されている属性が表示されます。

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 変更が許可されるには、セッションキーをトークンキーに変更する場合のみです。CloudHSM CLI の key set-attribute コマンドを使用して属性値を変更します。

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> この属性ではキー取得がサポートされています。すべてのパブリックキーで False を指定する必要があります。True に設定することはできません。シークレットキーまたは EC プライベートキーに対しては、True または False に設定できます。

- CKA_TRUSTED

> **Note**
>
> この属性は Crypto Officer (CO) のみによって True または False に設定できます。

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> この属性をエクスポート可能なデータキーに適用して、このキーを CKA_TRUSTED としてマークされたキーでのみラップできるように指定します。1度 CKA_WRAP_WITH_TRUSTED を true に設定すると属性は読み取り専用になり、属性を変更または削除することはできません。

AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリエラーコードの解釈

特定のキーでサポートされていない PKCS #11 ライブラリ属性をテンプレートで指定すると、エラーが発生します。次の表には、仕様に違反した場合に生成されるエラーコードが含まれています。

エラーコード	説明
CKR_TEMPLATE_INCONSISTENT	PKCS#11 仕様に準拠しているが、CloudHSM でサポートされていない属性を属性テンプレートで指定した場合に、このエラーが発生します。
CKR_ATTRIBUTE_TYPE_INVALID	PKCS#11 仕様に準拠しているが、CloudHSM でサポートされていない属性の値を取得すると、このエラーが発生します。
CKR_ATTRIBUTE_INCOMPLETE	このエラーは、属性テンプレートで必須属性を指定しなかった場合に発生します。
CKR_ATTRIBUTE_READ_ONLY	このエラーは、属性テンプレートで読み取り専用属性を指定した場合に発生します。

AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリのコードサンプル

GitHub のコードサンプルは、 AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリを使用して基本的なタスクを実行する方法を示しています。

前提条件

サンプルを実行する前に、以下のステップを実行して環境をセットアップします。

- クライアント SDK 5 用の PKCS #11 ライブラリ のインストールと設定をします。
- 暗号化ユーザー (CU) の設定をします。アプリケーションは、この HSM アカウントを使用して HSM でコードサンプルを実行します。

コードサンプル

PKCS#11 用 AWS CloudHSM ソフトウェアライブラリのコードサンプルは、GitHub で入手できます。このリポジトリには、暗号化、復号化、署名、検証など、PKCS #11 を使用して一般的な操作を行う方法の例が含まれています。

- キーの生成 (AES、RSA、EC)
- Ed25519 キーペアを生成する
- ML-DSA キーペアの生成
- キー属性のリスト化
- AES GCM を使用したデータの暗号化および復号
- AES_CTR を使用したデータの暗号化および復号
- 3DES を使用したデータの暗号化および復号
- RSAを使用したデータの署名と検証
- ECDSA でデータに署名して検証する
- EdDSA でデータに署名して検証する
- ML-DSA でデータに署名して検証する
- HMAC KDFを使用したキーの取得
- PKCS #5 パディングありの AES を使用したキーのラップとラップ解除
- パディングなしの AES を使用したキーのラップとラップ解除
- ゼロパディングありの AES を使用したキーのラップとラップ解除
- AES-GCM を使用したキーのラップとラップ解除
- RSA を使用したキーのラップとラップ解除

の PKCS #11 ライブラリの詳細設定 AWS CloudHSM

AWS CloudHSM PKCS #11 プロバイダーには、ほとんどのお客様が使用する一般的な設定の一部ではない、次の高度な設定が含まれています。これらの設定には追加機能があります。

- PKCS #11 による複数のスロットへの接続
- PKCS #11 の設定を再試行します

用の PKCS #11 ライブラリを使用した複数のスロット設定 AWS CloudHSM

クライアント SDK 5 PKCS #11 ライブラリ内の 1 つのスロットは、 AWS CloudHSM内のクラスターへの 1 つの接続を表します。クライアント SDK 5 では、1 つの PKCS #11 アプリケーションから複数のスロットでユーザーを複数の CloudHSM クラスターに接続できるように PKCS11 ライブラリを設定できます。

このトピックで説明されている手順に従って、アプリケーションがマルチスロット機能を使用して複数のクラスターに接続するようにします。

トピック

- の PKCS #11 ライブラリのマルチスロット前提条件 AWS CloudHSM
- のマルチスロット機能用に PKCS #11 ライブラリを設定する AWS CloudHSM
- のマルチスロット機能を持つクラスターを追加する AWS CloudHSM
- のマルチスロット機能を持つクラスターを削除する AWS CloudHSM

の PKCS #11 ライブラリのマルチスロット前提条件 AWS CloudHSM

PKCS #11 ライブラリの複数のスロットに を設定する前に AWS CloudHSM、次の前提条件を完了してください。

- 接続先の 2 つ以上の AWS CloudHSM クラスターとそのクラスター証明書。
- セキュリティグループが上記のすべてのクラスターに接続するように正しく設定された EC2 インスタンス。クラスターとクライアントインスタンスのセットアップ方法の詳細については、「 の開始方法 AWS CloudHSM」を参照してください。
- マルチスロット機能を設定するには、PKCS #11 ライブラリを事前にダウンロードしてインストールしておく必要があります。これをまだ確認していない場合は、「???」の手順を参照してください。

のマルチスロット機能用に PKCS #11 ライブラリを設定する AWS CloudHSM

のマルチスロット機能用に PKCS #11 ライブラリを設定するには AWS CloudHSM、次の手順に従います。

1. マルチスロット機能を使用して接続するクラスターを特定します。
2. ??? の手順に従って、これらのクラスターを PKCS #11 設定に追加します。
3. 次回 PKCS #11 アプリケーションを実行するときには、マルチスロット機能が使用できるようになります。

のマルチスロット機能を持つクラスターを追加する AWS CloudHSM

PKCS #11 for を使用して複数のスロットに接続する場合は AWS CloudHSM、 configure-pkcs11 add-cluster コマンドを使用して設定にクラスターを追加します。

構文

```
configure-pkcs11 add-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
        [--client-cert-hsm-tls-file <CLIENT CERTIFICATE FILE>]
        [--client-key-hsm-tls-file <CLIENT KEY FILE>]
        [-h, --help]
```

例

cluster-id パラメータを使用してクラスターを追加する

Example

configure-pkcs11 add-cluster とともに `cluster-id` パラメータを使用して、クラスター (`cluster-1234567` の ID) を設定に追加します。

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" add-cluster --
cluster-id <cluster-1234567>
```

> **Tip**
>
> configure-pkcs11 add-cluster を `cluster-id` パラメータと一緒に使用してもクラスターが追加されない場合は、以下の例を参照して、追加するクラスターを識別するための `--region` と `--endpoint` パラメータも必要な、より長いバージョンのこのコマンドを参照してください。例えば、クラスターのリージョンが AWS CLI のデフォルトとして設定されているものと異なる場合、適切なリージョンを使用するように `--region` パラメータを使用する必要があります。さらに、呼び出しに使用する AWS CloudHSM API エンドポイントを指定することもできます。これは、デフォルトの DNS ホスト名を使用しない VPC インターフェイスエンドポイントを使用するなど、さまざまなネットワーク設定に必要な場合があります AWS CloudHSM。

`cluster-id`、`endpoint`、および `region` パラメータを使用してクラスターを追加する

Example

configure-pkcs11 add-cluster とともに `cluster-id`、`endpoint`、`region` のパラメータを使用して、クラスター (`cluster-1234567` の ID) を設定に追加します。

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-id <cluster-1234567>
 --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" add-cluster --
cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-
east-1.amazonaws.com>
```

`--cluster-id`、`--region`、`--endpoint` パラメータの詳細については、the section called “パラメータ”を参照してください。

パラメータ

--cluster-id ***<Cluster ID>***

`DescribeClusters` を呼び出して、クラスターIDに関連付けられたクラスターのすべての HSM Elastic Network Interface (ENI) IPアドレスを検索します。システムは ENI IP アドレスを設定 AWS CloudHSM ファイルに追加します。

> **Note**
>
> パブリックインターネットにアクセスできない VPC 内の EC2 インスタンスから `--cluster-id`パラメータを使用する場合は、接続するインターフェイス VPC エンドポイントを作成する必要があります AWS CloudHSM。VPC エンドポイントの詳細については、「???」を参照してください。

必須: はい

--endpoint ***<Endpoint>***

`DescribeClusters` 呼び出しに使用する AWS CloudHSM API エンドポイントを指定します。このオプションは `--cluster-id` と組み合わせて設定する必要があります。

必須: いいえ

--hsm-ca-cert ***<HsmCA Certificate Filepath>***

HSM CA 証明書ファイルへのファイルパスを指定します。

必須: いいえ

--region ***<Region>***

クラスターのリージョンを指定します。このオプションは `--cluster-id` と組み合わせて設定する必要があります。

この `--region` パラメータを指定しない場合、システムは `AWS_DEFAULT_REGION` または `AWS_REGION` の環境変数の読み取りを試みてリージョンを選択します。これらの変数が設定されていない場合、環境変数で別のファイルを指定しない限り、AWS Config (通常は `~/.aws/config`) のプロファイルに関連付けられたリージョンをチェックします`AWS_CONFIG_FILE`。いずれも設定されていない場合は、`us-east-1` デフォルトでリージョンが設定されます。

必須: いいえ

--client-cert-hsm-tls-file ***<########## hsm tls ##>***

TLS クライアントと HSM の相互認証に使用するクライアント証明書へのパス。

このオプションは、CloudHSM CLI で HSM に少なくとも 1 つのトラストアンカーを登録している場合にのみ使用します。このオプションは `--client-key-hsm-tls-file` と組み合わせて設定する必要があります。

必須: いいえ

--client-key-hsm-tls-file ***<######## hsm tls ####***

TLS クライアントと HSM の相互認証に使用されるクライアントキーへのパス。

このオプションは、CloudHSM CLI で HSM に少なくとも 1 つのトラストアンカーを登録している場合にのみ使用します。このオプションは `--client-cert-hsm-tls-file` と組み合わせて設定する必要があります。

必須: いいえ

のマルチスロット機能を持つクラスターを削除する AWS CloudHSM

PKCS #11 を使用して複数のスロットに接続する場合は、configure-pkcs11 remove-cluster コマンドを使用して使用可能な PKCS#11 スロットからクラスターを削除します。

構文

```
configure-pkcs11 remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

例

`cluster-id` パラメータを使用してクラスターを削除します

Example

configure-pkcs11 remove-cluster とともに `cluster-id` パラメータを使用して、クラスター (`cluster-1234567` の ID) を設定から削除します。

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 remove-cluster --cluster-
id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" remove-cluster
 --cluster-id <cluster-1234567>
```

`--cluster-id` パラメータの詳細については、「the section called “パラメータ”」をご参照ください。

パラメータ

--cluster-id ***<Cluster ID>***

設定から削除するクラスターの ID

必須: はい

の PKCS #11 ライブラリのコマンドを再試行する AWS CloudHSM

AWS CloudHSM クライアント SDK 5.8.0 以降には、HSM スロットリングされたオペレーションをクライアント側から再試行する自動再試行戦略が組み込まれています。HSM が以前のオペレーションが多すぎてそれ以上リクエストを受け付けられないためにオペレーションをスロットリングすると、Client SDK はスロットリングされたオペレーションを最大 3 回再試行しますが、その間、エクスポネンシャルバックオフします。この自動再試行戦略は、オフとスタンダードの 2 つのモードのいずれかに設定できます。

- オフ: クライアント SDK は、HSM によってスロットリングされたオペレーションに対しては再試行戦略を一切実行しません。
- スタンダード: これはクライアント SDK 5.8.0 以降のデフォルトモードです。このモードでは、クライアント SDK はエクスポネンシャルバックオフすることで、スロットリングされた操作を自動的に再試行します。

詳細については、「HSM スロットリング」を参照してください。

再試行コマンドをオフモードに設定する

Linux

Linux でクライアント SDK 5 向けに再試行コマンドを off に設定するには

- 次のコマンドを使用して再試行設定を off モードに設定できます。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

Windows 上の クライアント SDK 5 向けに再試行コマンドを off に設定するには

- 次のコマンドを使用して再試行設定を off モードに設定できます。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --default-
retry-mode off
```

PKCS #11 ライブラリを使用した証明書ストレージ

AWS CloudHSM の PKCS #11 ライブラリは、hsm2m.medium クラスター上でパブリックキー証明書を「パブリックオブジェクト」(PKCS #11 2.40 で定義) として保存することをサポートしています。この機能により、パブリック PKCS #11 セッションとプライベート PKCS #11 セッションの両方で、パブリックキー証明書の作成、取得、変更、および削除が可能になります。

PKCS #11 ライブラリで証明書ストレージを使用するには、クライアント設定でこの機能を有効にする必要があります。有効化すると、PKCS #11 アプリケーションから証明書オブジェクトを管理できるようになります。証明書とキーの両方に適用されるオペレーション (C_FindObjects など) は、キーと証明書の両方のストレージから結果を返します。

> **証明書ストレージの制限**
>
> 証明書ストレージは、各クラスターの保存された証明書の数と、各 HSM の読み取りおよび書き込みオペレーションのレートを制限します。詳細については、「証明書ストレージの制限」を参照してください。

トピック

- 証明書ストレージの有効化
- 証明書ストレージ API オペレーション
- 証明書ストレージの属性
- 証明書ストレージの監査ログ
- 証明書ストレージの制限

証明書ストレージの有効化

PKCS #11 ライブラリの設定ツールを使用して、hsm2m.medium クラスターで証明書ストレージを有効化できます。この機能は、SDK バージョン 5.13 以降で利用可能です。証明書オブジェクトタイプをサポートするオペレーションの一覧については、証明書ストレージ API オペレーション を参照してください。

証明書ストレージを有効にするには、使用しているオペレーティングシステムに応じて次の手順に従ってください。

Linux

- 証明書ストレージを有効にする

  次のコマンドを実行します。

  ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --enable-certificate-storage
  ```

Windows

- 証明書ストレージを有効にする

  コマンドプロンプトを開き、次のコマンドを実行します。

  ```
  PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --enable-
  certificate-storage
  ```

証明書ストレージ API オペレーション

次の PKCS #11 オペレーションは、証明書オブジェクトタイプ (CKO_CERTIFICATE) をサポートしています。

一般的な証明書オペレーション

C_CreateObject

新しい証明書オブジェクトを作成します。

C_DestroyObject

既存の証明書オブジェクトを削除します。

C_GetAttributeValue

証明書オブジェクトの 1 つ以上の属性の値を取得します。

C_SetAttributeValue

証明書オブジェクトの 1 つ以上の属性の値を更新します。

証明書オブジェクトの検索オペレーション

C_FindObjectsInit

証明書オブジェクトの検索を開始します。

C_FindObjects

証明書オブジェクトの検索を続行します。

C_FindObjectsFinal

証明書オブジェクトの検索を終了します。

リターンコードのスロットリング

証明書ストレージが読み取りまたは書き込みのレート制限を超えるオペレーションを調整する場合、リターンコードはオペレーションによって異なります。書き込みオペレーション (C_CreateObject、C_SetAttributeValue、および C_DestroyObject) とC_FindObjectsInit読み取りオペレーションは を返しますCKR_FUNCTION_FAILED。

C_GetAttributeValue は現在 を返しますCKR_DEVICE_ERROR。いずれの場合も、 でスロットリングを処理するためのベストプラクティスに従ってくださいHSM スロットリング。

証明書ストレージの属性

次の表に、サポートされている証明書オブジェクトの属性とその値を示します。

属性	デフォルトの値	説明
CKA_CLASS	必須	CKO_CERTIFICATE を指定してください。
CKA_TOKEN	正	True を指定してください。
CKA_MODIFIABLE	正	True を指定してください。
CKA_PRIVATE	誤	False を指定してください。
CKA_LABEL	空	最大 127 文字。
CKA_COPYABLE	誤	False を指定してください。
CKA_DESTROYABLE	正	True を指定してください。
CKA_CERTIFICATE_TYPE	必須	CKC_X_509 を指定してください。
CKA_TRUSTED	誤	False を指定してください。
CKA_CERTIFICATE_CATEGORY	CK_CERTIFICATE_CATEGORY_UNSPECIFIED	CK_CERTIFICATE_CATEGORY_UNSPECIFIED を指定してください。
CKA_CHECK_VALUE	CKA_VALUE から派生	CKA_VALUE に基づいて自動的に設定されます。
CKA_START_DATE	空	証明書の「Not Before」日付。
CKA_END_DATE	空	証明書の「Not After」日付。

属性	デフォルトの値	説明
CKA_PUBLIC_KEY_INFO	空	最大サイズは 16 キロバイトです。
CKA_SUBJECT	必須	証明書のサブジェクト。
CKA_ID	空	最大サイズは 128 バイトです。一意性は強制されません。
CKA_ISSUER	空	証明書の発行者。
CKA_SERIAL_NUMBER	空	証明書のシリアル番号。
CKA_VALUE	必須	最大サイズは 32 キロバイトです。

証明書ストレージの監査ログ

AWS CloudHSM は、データを変更する証明書ストレージのオペレーションについて、クラスターの CloudWatch ロググループ内にある Amazon CloudWatch Events の個別ログストリームに監査ログを書き込みます。このログストリームは、クラスター単位で命名され、クラスタ内の特定の HSM ごとには作成されません。

CloudWatch で監査ログにアクセスする方法については、Amazon CloudWatch Logs と AWS CloudHSM 監査ログの使用 を参照してください。

ログエントリフィールド

`object_handle`

証明書オブジェクトの一意の識別子。

`op_code`

実行または試行されたオペレーション。使用できる値:

- `CreateObject`
- `DestroyObject`
- `SetAttributeValues`

`response`

オペレーションが成功した場合は OK、または次のいずれかのエラータイプ。

- `DuplicateAttribute`
- `InvalidAttributeValue`
- `ObjectNotFound`
- `MaxObjectsReached`
- `InternalFailure`

`attributes`

変更された属性 (存在する場合)。

`timestamp`

オペレーションが発生した時刻 (Unix エポックからの経過ミリ秒)。

監査ログの例

CreateObject の例

```
{
    "object_handle": 463180677312929947,
    "op_code": "CreateObject",
    "response": "OK",
    "attributes": null,
    "timestamp": 1725482483671
}
```

DestroyObject の例

```
{
    "object_handle": 463180677312929947,
    "op_code": "DestroyObject",
    "response": "OK",
    "attributes": null,
    "timestamp": 1725482484559
}
```

SetAttributeValues の例

```
{
    "object_handle": 463180678453346687,
    "op_code": "SetAttributeValues",
    "response": "OK",
    "attributes": [
        "Label"
    ],
    "timestamp": 1725482488004
}
```

失敗した CreateObject の例

```
{
    "object_handle": null,
    "op_code": "CreateObject",
    "response": "MaxObjectsReached",
    "attributes": null,
    "timestamp": 1726084937125
}
```

証明書ストレージの制限

証明書ストレージには、次の制限が適用されます。これらの制限は変更できません。各クラスターに適用されるストアド証明書の最大数。読み取りと書き込みのレート制限は各 HSM に適用されるため、HSMs。

制限	値	スコープ
保存された証明書の最大数	60	各クラスター
1 秒あたりの読み取りオペレーション	10	各 HSM
1 秒あたりの書き込みオペレーション	1	各 HSM

読み取りオペレーションには、 C_FindObjectsInitおよび が
含まれますC_GetAttributeValue。書き込みオペレーションに
は、C_CreateObject、C_SetAttributeValue、および が含まれますC_DestroyObject。

これらの制限を超えると、次の動作が適用されます。

- 保存されている証明書の最大数を超えると、C_CreateObjectオペレーションは を返しCKR_FUNCTION_FAILED、証明書ストレージは証明書ストレージ監査ログにMaxObjectsReachedエラーを記録します。既存の証明書は読み取り可能です。追加の証明書を保存するには、不要になった証明書を削除します。
- 読み取りまたは書き込みのレート制限を超えると、証明書ストレージはリクエストをスロットリングしC_GetAttributeValue、 オペレーションは を返します。ただしCKR_FUNCTION_FAILED、 は を返します CKR_DEVICE_ERROR (「」を参照問題: スロットリングCKR_DEVICE_ERROR時に C_GetAttributeValue が返されます)。PKCS #11 ライブラリはスロットリングされた証明書ストレージオペレーションを再試行しないため (「」を参照問題: PKCS #11 ライブラリがスロットリングされた証明書ストレージオペレーションを再試行しない)、アプリケーションはエクスポネンシャルバックオフで再試行する必要があります。詳細については、「HSM スロットリング」を参照してください。

読み取りレートの制限内に収まるようにするには、アプリケーションが送信する読み取りオペレーションの数を減らします。各C_GetAttributeValue呼び出しは、テンプレート内の属性の数に関係なく、1 回の読み取りオペレーションとしてカウントされます。1 回の呼び出しで必要なすべての属性をリクエストするアプリケーションは、1 回の読み取りオペレーションを使用します。属性ごとに個別の呼び出しを行うアプリケーションは、各属性に対して 1 つの読み取りオペレーションを使用します。HSMs を追加すると、クラスターがサポートする合計レートが増加しますが、この比率は変更されません。詳細については、「AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリを使用して属性を取得する」を参照してください。

AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine

AWS CloudHSM OpenSSL Dynamic Engine を使用すると、OpenSSL API を使用して暗号化オペレーションを CloudHSM OpenSSL クラスターにオフロードできます。

AWS CloudHSM には OpenSSL Dynamic Engine が用意されています。このエンジンは、 AWS CloudHSM JSSE で Tomcat を使用した Linux での SSL/TLS オフロードまたは で確認できますAWS CloudHSM NGINX または Apache with OpenSSL を使用した Linux での SSL/TLS オフロード。OpenSSL AWS CloudHSM で を使用する例については、この AWS セキュリティブログを参照してください。SDK のプラットフォームサポートトの詳細については、「the section called “サポートされているプラットフォーム”」を参照してください。トラブルシューティングについては、「の OpenSSL Dynamic Engine の既知の問題 AWS CloudHSM」を参照してください。

クライアント SDK 5 を使用して OpenSSL の AWS CloudHSM 動的エンジンをインストールおよび設定するには、以下のセクションを使用します。

クライアント SDK 3 の使用の詳細については、「以前の SDK バージョンを使用して を操作する AWS CloudHSM」を参照してください。

トピック

- AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine をインストールする
- OpenSSL Dynamic Engine for AWS CloudHSM Client SDK 5 でサポートされているキータイプ
- OpenSSL Dynamic Engine for AWS CloudHSM Client SDK 5 でサポートされているメカニズム
- OpenSSL for の詳細設定 AWS CloudHSM

AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine をインストールする

以下のセクションを使用して、 AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine をインストールします。

> **Note**
>
> クライアント SDK 5 で単一の HSM クラスターを実行するには、まず `disable_key_availability_check` を `True` に設定してクライアントキーの耐久性の設定を管理する必要があります。詳細については、キーの同期 と クライアント SDK 5 設定ツール を参照してください。

OpenSSL Dynamic Engine をインストールして設定するには

1. 以下のコマンドを使用して、OpenSSL エンジンをダウンロードしてインストールします。

   Amazon Linux 2023

   Amazon Linux 2023 用 OpenSSL Dynamic Engine を X86_64 アーキテクチャにインストールします。

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
   cloudhsm-dyn-latest.amzn2023.x86_64.rpm
   ```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.x86_64.rpm
```

Amazon Linux 2023 用 OpenSSL Dynamic Engine を ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

Amazon Linux 2 用 OpenSSL Dynamic Engine を x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
```

Amazon Linux 2 用 OpenSSL Dynamic Engine を ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
dyn-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.aarch64.rpm
```

RHEL 9 (9.2+)

RHEL 9 用 OpenSSL Dynamic Engine を X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-
dyn-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el9.x86_64.rpm
```

RHEL 9 用 OpenSSL Dynamic Engine を ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-dyn-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

RHEL 8 用 OpenSSL Dynamic Engine を X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-dyn-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el8.x86_64.rpm
```

ARM64 アーキテクチャに RHEL 8 用の OpenSSL Dynamic Engine をインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-dyn-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el8.aarch64.rpm
```

Ubuntu 26.04 LTS

OpenSSL Dynamic Engine for Ubuntu 26.04 LTS を x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/cloudhsm-dyn_latest_u26.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u26.04_amd64.deb
```

OpenSSL Dynamic Engine for Ubuntu 26.04 LTS を ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/
cloudhsm-dyn_latest_u26.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u26.04_arm64.deb
```

Ubuntu 24.04 LTS

OpenSSL Dynamic Engine for Ubuntu 24.04 LTS を x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/
cloudhsm-dyn_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u24.04_amd64.deb
```

OpenSSL Dynamic Engine for Ubuntu 24.04 LTS を ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/
cloudhsm-dyn_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS 用 OpenSSL Dynamic Engine を X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-dyn_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_amd64.deb
```

Ubuntu 22.04 LTS 用 OpenSSL Dynamic Engine をARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-dyn_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_arm64.deb
```

ダイナミックエンジン用の共有ライブラリが /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so にインストールされました。

2. クライアント SDK 5 をブートストラップします。ブートストラップの詳細については、クライアント SDK をブートストラップする を参照してください。
3. Crypto User (CU) の認証情報を使用して環境変数を設定します。CU の詳細については、「the section called “作成”」を参照してください。

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

> **Note**
>
> クライアント SDK 5 では CU の認証情報を保存するための CLOUDHSM_PIN 環境変数が導入されています。クライアント SDK 3 では、CU の認証情報を n3fips_password 環境変数に保存します。クライアント SDK 5 は両方の環境変数をサポートしますが、CLOUDHSM_PIN を使用することを推奨します。
> CLOUDHSM_PIN 環境変数を設定するときは、シェルによって解釈される可能性のある特殊文字をエスケープする必要があります。

4. OpenSSL Dynamic Engine のインストールをクラスターに接続します。詳細については、クラスターへの接続 を参照してください。
5. クライアント SDK 5 をブートストラップします。詳細については、「the section called “クライアント SDK をブートストラップする”」を参照してください。

クライアント SDK 5 用 [OpenSSL 動的エンジン] を確認します。

次のコマンドを使用して [OpenSSL 動的エンジン] のインストールを確認します。

```
$ openssl engine -t cloudhsm
```

次の出力により設定が検証されます。

```
(cloudhsm) CloudHSM OpenSSL Engine
     [ available ]
```

OpenSSL Dynamic Engine for AWS CloudHSM Client SDK 5 でサポートされているキータイプ

AWS CloudHSM OpenSSL Dynamic Engine は、クライアント SDK 5 で次のキータイプをサポートしています。

キータイプ	説明
EC	P-256、P-384、および secp256k1 キータイプの ECDSA 署名/検証。OpenSSL エンジンと相互運用可能な EC キーを生成するには、CloudHSM CLI で非対称キーをエクスポートする を参照してください。
RSA	2,048、3,072、および 4,096 ビットキーの RSA キーの生成および RSA 署名/検証 検証は OpenSSL ソフトウェアにオフロードされます。

OpenSSL Dynamic Engine for AWS CloudHSM Client SDK 5 でサポートされているメカニズム

AWS CloudHSM OpenSSL Dynamic Engine は、クライアント SDK 5 で関数に署名および検証するための以下のメカニズムをサポートしています。

署名および検証

クライアント SDK 5 では、データはソフトウェアでローカルにハッシュされます。これは、ハッシュできるデータのサイズに制限がないことを意味します。

RSA 署名タイプ

- SHA1withRSA
- SHA224withRSA

- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

ECDSA 署名タイプ

- SHA1 (ECDSA 搭載)
- SHA224 (ECDSA 搭載)
- SHA256 (ECDSA 搭載)
- SHA384 (ECDSA 搭載)
- SHA512 (ECDSA 搭載)

OpenSSL for の詳細設定 AWS CloudHSM

AWS CloudHSM OpenSSL プロバイダーには次の詳細設定が含まれていますが、これはほとんどのお客様が使用する一般的な設定の一部ではありません。これらの設定には追加機能があります。

- OpenSSL の再試行コマンド

の OpenSSL の再試行コマンド AWS CloudHSM

AWS CloudHSM クライアント SDK 5.8.0 以降には、HSM スロットリングされたオペレーションをクライアント側から再試行する自動再試行戦略が組み込まれています。HSM が以前のオペレーションが多すぎてそれ以上リクエストを受け付けられないためにオペレーションをスロットリングすると、Client SDK はスロットリングされたオペレーションを最大 3 回再試行しますが、その間、エクスポネンシャルバックオフします。この自動再試行戦略は、オフとスタンダードの 2 つのモードのいずれかに設定できます。

- オフ: クライアント SDK は、HSM によってスロットリングされたオペレーションに対しては再試行戦略を一切実行しません。
- スタンダード: これはクライアント SDK 5.8.0 以降のデフォルトモードです。このモードでは、クライアント SDK はエクスポネンシャルバックオフすることで、スロットリングされた操作を自動的に再試行します。

詳細については、「HSM スロットリング」を参照してください。

再試行コマンドをオフモードに設定する

再試行コマンドを off モードに設定するには以下のコマンドを使用できます。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

AWS CloudHSM クライアント SDK 5 の OpenSSL プロバイダー

AWS CloudHSM OpenSSL プロバイダーを使用すると、OpenSSL プロバイダー API を使用して TLS 暗号化オペレーションを CloudHSM OpenSSL クラスターにオフロードできます。プロバイダーインターフェイスは、OpenSSL 3.2 以降を使用する新しいデプロイに推奨されるアプローチです。

クライアント SDK 5 を使用して AWS CloudHSM OpenSSL プロバイダーをインストールおよび設定するには、以下のセクションを使用します。

> ML-DSA の OpenSSL バージョン要件
>
> ML-DSA キータイプには OpenSSL 3.5 以降が必要です。

サポートされているプラットフォーム

OpenSSL プロバイダーには、EL9+、EL10+、Ubuntu 26.04 LTS、および Amazon Linux 2023+ で利用可能な OpenSSL 3.2 以降が必要です。

互換性を検証します。 `openssl version`

トピック

- AWS CloudHSM クライアント SDK 5 用の OpenSSL プロバイダーをインストールする
- AWS CloudHSM クライアント SDK 5 の OpenSSL プロバイダーでサポートされているキータイプ
- OpenSSL プロバイダーがサポートするメカニズム
- OpenSSL プロバイダーの詳細設定

AWS CloudHSM クライアント SDK 5 用の OpenSSL プロバイダーをインストールする

以下のセクションを使用して、AWS CloudHSM Client SDK 5 用の OpenSSL プロバイダーをインストールします。

> **Note**
>
> クライアント SDK 5 で単一の HSM クラスターを実行するには、まず `disable_key_availability_check` を `True` に設定してクライアントキーの耐久性の設定を管理する必要があります。詳細については、キーの同期 と クライアント SDK 5 設定ツール を参照してください。

要件

OpenSSL プロバイダーには、hsm2m.medium クラスタータイプと最小 CloudHSM Client SDK バージョン 5.17.0 以降が必要です。

OpenSSL プロバイダーをインストールする

OpenSSL プロバイダーをインストールするには

1. 次のコマンドを使用して、OpenSSL プロバイダーをダウンロードしてインストールします。

   Amazon Linux 2023

   Amazon Linux 2023 用の OpenSSL プロバイダーを x86_64 アーキテクチャにインストールします。

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
   cloudhsm-openssl-provider-latest.amzn2023.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-openssl-provider-latest.amzn2023.x86_64.rpm
   ```

   Amazon Linux 2023 用の OpenSSL プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-openssl-provider-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.amzn2023.aarch64.rpm
```

RHEL 9 (9.2+)

RHEL 9 用の OpenSSL プロバイダーを x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-
openssl-provider-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el9.x86_64.rpm
```

ARM64 アーキテクチャに RHEL 9 用の OpenSSL プロバイダーをインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-
openssl-provider-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el9.aarch64.rpm
```

RHEL 10 (10.0+)

RHEL 10 用の OpenSSL プロバイダーを x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/
cloudhsm-openssl-provider-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el10.x86_64.rpm
```

ARM64 アーキテクチャに RHEL 10 用の OpenSSL プロバイダーをインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/
cloudhsm-openssl-provider-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el10.aarch64.rpm
```

Ubuntu 26.04 LTS

OpenSSL Provider for Ubuntu 26.04 LTS を x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/
cloudhsm-openssl-provider_latest_u26.04_amd64.deb
```

```
$ sudo dpkg -i ./cloudhsm-openssl-provider_latest_u26.04_amd64.deb
```

OpenSSL Provider for Ubuntu 26.04 LTS を ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/
cloudhsm-openssl-provider_latest_u26.04_arm64.deb
```

```
$ sudo dpkg -i ./cloudhsm-openssl-provider_latest_u26.04_arm64.deb
```

OpenSSL プロバイダーの共有ライブラリが にインストールされています`/opt/cloudhsm/lib/licloudhsm_openssl_provider.so`。

2. クライアント SDK 5 をブートストラップします。ブートストラップの詳細については、クライアント SDK をブートストラップする を参照してください。
3. Crypto User (CU) 認証情報を使用して `CLOUDHSM_PIN` 環境変数を設定します。

```
$ export CLOUDHSM_PIN=<username>:<password>
```

4. OpenSSL プロバイダーのインストールをクラスターに接続します。詳細については、クラスターへの接続 を参照してください。

インストールの検証

OpenSSL プロバイダーが正しくインストールされていることを確認します。

```
$ CLOUDHSM_PIN=<username>:<password> openssl list -providers -provider cloudhsm
```

次のような出力が表示されます:

```
Providers:
  cloudhsm
    name: AWS CloudHSM OpenSSL Provider
    version: 5.17.0
    status: active
  default
    name: OpenSSL Default Provider
    version: 3.2.2
    status: active
```

AWS CloudHSM クライアント SDK 5 の OpenSSL プロバイダーでサポートされているキータイプ

AWS CloudHSM OpenSSL プロバイダーは、クライアント SDK 5 で次のキータイプをサポートしています。

キータイプ	説明
RSA	RSA 署名/検証および非対称暗号化オペレーション。検証は OpenSSL ソフトウェアにオフロードされます。OpenSSL プロバイダーと相互運用可能な RSA キーを生成するには、「」を参照してくださいCloudHSM CLI で非対称キーをエクスポートする。
EC	P-256、P-384、P-521 曲線の ECDSA 署名/検証。検証は OpenSSL ソフトウェアにオフロードされます。OpenSSL プロバイダーと相互運用可能な EC キーを生成するには、「」を参照してくださいCloudHSM CLI で非対称キーをエクスポートする。
Ed25519	Curve25519 (RFC 8032) を使用した EdDSA 署名/検証オペレーション。検証は OpenSSL ソフトウェアにオフロードされます。Ed25519 は、FIPS 以外のクラスターでのみ使用できます。OpenSSL プロバイダーと相互運用可能な Ed25519 キーを生成するには、「」を参照し

キータイプ	説明
	てくださいCloudHSM CLI で非対称キーをエクスポートする。
ML-DSA-44、ML-DSA-65、ML-DSA-87	FIPS 204 で定義されている量子後デジタル署名署名/検証オペレーション。検証は OpenSSL ソフトウェアにオフロードされます。ML-DSA[1] には OpenSSL 3.5 以降が必要です。OpenSSL プロバイダーと相互運用可能な ML-DSA キーを生成するには、「」を参照してくださいCloudHSM CLI で非対称キーをエクスポートする。

[1] 2026 年 9 月 1 日以降、ML-DSA は hsm2m.medium クラスターの FIPS モードで使用できます。

OpenSSL プロバイダーがサポートするメカニズム

AWS CloudHSM OpenSSL プロバイダー SDK は、デジタル署名、非対称暗号化、対称暗号化、キー交換など、さまざまなオペレーションのための包括的な暗号化メカニズムのセットをサポートしています。

RSA 署名タイプ

OpenSSL プロバイダーは、複数のハッシュアルゴリズムとパディングスキームを持つ RSA デジタル署名をサポートしています。

SHA1withRSA

SHA-1 ハッシュアルゴリズムを使用した RSA 署名

- PKCS#1 v1.5 パディング
- PSS (確率的署名スキーム) パディング

SHA224withRSA

SHA-224 ハッシュアルゴリズムを使用した RSA 署名

- PKCS#1 v1.5 パディング
- PSS パディング

SHA256withRSA

SHA-256 ハッシュアルゴリズムを使用した RSA 署名

- PKCS#1 v1.5 パディング
- PSS パディング

SHA384withRSA

SHA-384 ハッシュアルゴリズムを使用した RSA 署名

- PKCS#1 v1.5 パディング
- PSS パディング

SHA512withRSA

SHA-512 ハッシュアルゴリズムを使用した RSA 署名

- PKCS#1 v1.5 パディング
- PSS パディング

ECDSA 署名タイプ

OpenSSL プロバイダーは、複数のハッシュアルゴリズムを持つ ECDSA デジタル署名をサポートしています。

SHA1 (ECDSA 搭載)

SHA-1 ハッシュアルゴリズムを使用した ECDSA 署名

SHA224 (ECDSA 搭載)

SHA-224 ハッシュアルゴリズムを使用した ECDSA 署名

SHA256 (ECDSA 搭載)

SHA-256 ハッシュアルゴリズムを使用した ECDSA 署名

SHA384 (ECDSA 搭載)

SHA-384 ハッシュアルゴリズムを使用した ECDSA 署名

SHA512 (ECDSA 搭載)

SHA-512 ハッシュアルゴリズムを使用した ECDSA 署名

EdDSA 署名タイプ

OpenSSL プロバイダーは EdDSA デジタル署名をサポートしています。Ed25519 は、FIPS 以外のクラスターでのみ使用できます。

Ed25519

Ed25519 は、ワンショット署名と内部 SHA-512 ハッシュを使用して Curve25519 (RFC 8032) を使用して EdDSA 署名を実行します。外部ダイジェストアルゴリズムを選択することはできません。Ed25519 は TLS 1.3 署名ネゴシエーションをサポートしています。

ML-DSA 署名タイプ

OpenSSL プロバイダーは、FIPS 204 で定義されているように、ML-DSA (Module-Lattice Digital Signature Algorithm) ポスト量子デジタル署名をサポートしています。ML-DSA には OpenSSL 3.5 以降が必要です。

ML-DSA-44、ML-DSA-65、ML-DSA-87

ML-DSA は、NIST セキュリティレベル 2 (ML-DSA-44、128 ビット）、3 (ML-DSA-65、192 ビット）、および 5 (ML-DSA-87、256 ビット) でポスト量子署名を提供します。すべてのバリアントは、ワンショットの純粋モード署名を使用します。外部ダイジェストアルゴリズムを選択することはできません。ML-DSA は TLS 1.3 署名ネゴシエーションをサポートしています。

OpenSSL プロバイダーの詳細設定

AWS CloudHSM OpenSSL プロバイダー SDK には、さまざまなユースケースや環境に合わせて動作をカスタマイズするための高度な設定オプションが用意されています。設定ツールを使用して、これらのオプションを設定します。

- OpenSSL プロバイダーの設定を再試行する

の OpenSSL プロバイダーの再試行コマンド AWS CloudHSM

AWS CloudHSM クライアント SDK 5.8.0 以降には、HSM スロットリングされたオペレーションをクライアント側から再試行する自動再試行戦略が組み込まれています。HSM が以前のオペレーションが多すぎてそれ以上リクエストを受け付けられないためにオペレーションをスロットリングすると、Client SDK はスロットリングされたオペレーションを最大 3 回再試行しますが、その間、エク

スポネンシャルバックオフします。この自動再試行戦略は、オフとスタンダードの 2 つのモードのいずれかに設定できます。

- オフ: クライアント SDK は、HSM によってスロットリングされたオペレーションに対しては再試行戦略を一切実行しません。
- スタンダード: これはクライアント SDK 5.8.0 以降のデフォルトモードです。このモードでは、クライアント SDK はエクスポネンシャルバックオフすることで、スロットリングされた操作を自動的に再試行します。

詳細については、「HSM スロットリング」を参照してください。

再試行コマンドをオフモードに設定する

Linux

Linux でクライアント SDK 5 向けに再試行コマンドを off に設定するには

- 次のコマンドを使用して再試行設定を off モードに設定できます。

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --default-retry-mode off
```

AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP)

キーストレージプロバイダー (KSP) は、Microsoft Windows オペレーティングシステム固有の暗号 API です。キーストレージプロバイダー (KSP) を使用すると、開発者は暗号化技術を使用して Windows ベースのアプリケーションを保護できます。

ブートストラップの詳細については、「クラスターへの接続」を参照してください。

Client SDK 3 の使用の詳細については、「以前の SDK バージョンを使用して を操作する AWS CloudHSM」を参照してください。

トピック

- AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) をインストールする
- AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) への認証
- AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) でサポートされているキータイプ

- サポートされている API オペレーション AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP)
- の KSP の詳細設定 AWS CloudHSM

AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) をインストールする

以下のセクションを使用して、 AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) をインストールします。

> **Note**
>
> クライアント SDK 5 で単一の HSM クラスターを実行するには、まず `disable_key_availability_check` を `True` に設定してクライアントキーの耐久性の設定を管理する必要があります。詳細については、キーの同期 と クライアント SDK 5 設定ツール を参照してください。

キーストレージプロバイダー (KSP) をインストールして設定するには

1. Windows Server 用のキーストレージプロバイダー (KSP) を x86_64 アーキテクチャにインストールし、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMKSP-latest.msi -Outfile C:\AWSCloudHSMKSP-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMKSP-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

2. 構成ツールを使用して、証明書の発行場所を指定します。手順については、「発行証明書の場所を指定する」を参照してください。
3. クラスターに接続して使用するには、「クライアント SDK をブートストラップする」を参照してください。
4. キーストレージプロバイダー (KSP) ファイルは、次の場所にあります。
   - Windows のバイナリ:

```
C:\Program Files\Amazon\CloudHSM
```

Windows の設定スクリプトとログファイル:

```
C:\ProgramData\Amazon\CloudHSM
```

AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) への認証

AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) を使用する前に、システムで HSM のログイン認証情報を設定する必要があります。これには 2 つのオプションがあります。

- Windows 資格情報マネージャー (より高いセキュリティのため推奨)
- システム環境変数 (より簡単なセットアップ)

Windows 資格情報マネージャー

認証情報は、`set_cloudhsm_credentials` ユーティリティまたは Windows 資格情報マネージャーのインターフェイスを使用して設定できます。

- **`set_cloudhsm_credentials` ユーティリティの使用:**

  Windows インストーラには `set_cloudhsm_credentials` ユーティリティが含まれています。このユーティリティを使用して、HSM ログイン認証情報を Windows 資格情報マネージャーに簡単に渡すことができます。このユーティリティをソースからコンパイルする場合は、インストーラに含まれている Python コードを使用できます。

  1. `C:\Program Files\Amazon\CloudHSM\tools\` に移動します。
  2. 次のコマンドを実行します。

  ```
  set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
  ```

- 資格情報マネージャーのインターフェイスの使用:

  1. 認証情報マネージャーを開く:
     - タスクバーの検索ボックスに `credential manager` と入力します
     - [資格情報マネージャー] を選択します

2. [Windows 資格情報] を選択して、Windows 資格情報を管理します。
3. [汎用資格情報を追加] を選択します
4. 次の詳細情報を入力します。
   - インターネットまたはネットワークアドレス: `CLOUDHSM_PIN`。
   - ユーザー名: *<CU USER>*。
   - パスワード: *<CU PASSWORD>*。
5. [OK] をクリックします。

システム環境変数

ご使用の HSM および Crypto User (CU) を識別するシステム環境変数を設定できます。

> **Warning**
>
> システム環境変数を使用して認証情報を設定すると、パスワードがシステム上にプレーンテキストで保存されます。セキュリティを向上させるには、代わりに Windows 資格情報マネージャーを使用してください。

以下のいずれかを使用して環境変数を設定できます。

- setx。
- Windows の [システムプロパティ] コントロールパネル ([詳細設定] タブ)。
- プログラムによるシステム環境変数の永続的な設定方法。

システム環境変数を設定する方法

`CLOUDHSM_PIN=`*`<CU USERNAME>`*`:`*`<CU PASSWORD>`*

HSM の Crypto User (CU) を識別し、必要なすべてのログイン情報を提供します。アプリケーションはこの CU として認証および実行します。このアプリケーションには、この CU のアクセス権限があり、CU が所有および共有しているキーのみを表示および管理できます。新しい CU を作成するには、CloudHSM CLI でuser create コマンドを使用します。既存の CU を検索するには、CloudHSM CLI で user list コマンドを使用します。

例えば、次のようになります。

```
setx /m CLOUDHSM_PIN test_user:password123
```

> **Note**
>
> CLOUDHSM_PIN 環境変数を設定するときは、シェルによって解釈される可能性のある特殊文字をエスケープする必要があります。

AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) でサポートされているキータイプ

AWS CloudHSM キーストレージプロバイダー (KSP) は、クライアント SDK 5 で次のキータイプをサポートしています。

キータイプ	説明
EC	secp256r1 (P-256)、secp384r1 (P-384)、secp521r1 (P-521) 曲線を使用してキーを生成します。
RSA	2,048 ビット、3,072 ビット、および 4,096 ビットの RSA キーを生成します。

サポートされている API オペレーション AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP)

KSP のパラメータは、Microsoft KSP によって定義されます。詳細については、Microsoft のドキュメントを参照してください。

キーストレージプロバイダー (KSP) は、 AWS CloudHSM クライアント SDK 5 の次の KSP API オペレーションをサポートしています。

- NCryptOpenStorageProvider
- NCryptOpenKey
- NCryptCreatePersistedKey
- NCryptGetProperty

- NCryptSetProperty
- NCryptFinalizeKey
- NCryptDeleteKey
- NCryptFreeObject
- NCryptFreeBuffer
- NCryptIsAlgSupported
- NCryptEnumAlgorithms
- NCryptEnumKeys
- NCryptExportKey
- NCryptSignHash
- NCryptVerifySignature

キーストレージプロバイダー (KSP) を使用した NCryptOpenStorageProvider 関数

`NCryptOpenStorageProvider` 関数は、キーストレージプロバイダー (KSP) をロードして初期化します。

パラメータ

`phProvider` [out]

プロバイダーハンドルを保存する `NCRYPT_PROV_HANDLE` 変数へのポインタ。

`pszProviderName` [in]

キーストレージプロバイダーを識別する、null 終端された Unicode 文字列へのポインタ。AWS CloudHSM キーストレージプロバイダー (KSP) は以下の値をサポートしています。

値	意味
L"CloudHSM Key Storage Provider"	クライアント SDK 5 プロバイダー名を識別します。デフォルトでは、この名前を使用することをお勧めします。
L"Cavium Key Storage Provider"	クライアント SDK 3 プロバイダー名を識別します。下位互換性のサポート

> **Note**
>
> 値は、リテラルの前に L が付いていることからわかるように、ワイド文字列リテラルです。

`dwFlags` [in]

関数の動作を変更するフラグ。この関数にはフラグが定義されていません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。

キーストレージプロバイダー (KSP) を使用した NCryptOpenKey

`NCryptOpenKey` 関数は、キーストレージプロバイダー (KSP) に存在するキーを開きます。

パラメータ

`hProvider` [in]

キーを含む KSP ハンドル。このハンドルは、`NCryptOpenStorageProvider` を使用して取得します。

`phKey` [out]

キーハンドルを保存する `NCRYPT_KEY_HANDLE` 変数へのポインタ。

`pszKeyName` [in]

キー名を含む、null 終端された Unicode 文字列へのポインタ。

dwLegacyKeySpec [in, unused]

AWS CloudHSM Key Storage Provider (KSP) はこのパラメータを使用しません。

dwFlags [in]

関数の動作を変更するフラグ。この関数にはフラグが定義されていません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_INVALID_HANDLE	hProvider のハンドルが無効です。
NTE_BAD_KEYSET	指定されたキー名は一意の結果を返しませんでした。

キーストレージプロバイダー (KSP) を使用した NCryptCreatePersistedKey

NCryptCreatePersistedKey 関数は、新しいキーを作成し、それをキーストレージプロバイダー (KSP) に保存します。キーを作成した後、NCryptSetProperty 関数を使用してそのプロパティを設定できます。キーを使用する前に、NCryptFinalizeKey を呼び出す必要があります。

パラメータ

hProvider [in]

キーを作成するキーストレージプロバイダーのハンドル。このハンドルは、NCryptOpenStorageProvider を使用して取得します。

phKey [out]

キーハンドルを収納する NCRYPT_KEY_HANDLE 変数のアドレス。

pszAlgId [in]

キーを作成するための暗号アルゴリズム識別子を指定する、null 終端された Unicode 文字列へのポインタ。

AWS CloudHSM Key Storage Provider (KSP) は、次のアルゴリズムをサポートしています。

定数/値。	説明
BCRYPT_RSA_ALGORITHM "RSA"	RSA パブリックキーアルゴリズム。
BCRYPT_ECDSA_P256_ALGORITHM "ECDSA_P256"	256 ビット素数曲線デジタル署名アルゴリズム (FIPS 186-2)。
BCRYPT_ECDSA_P384_ALGORITHM "ECDSA_P384"	384 ビット素数曲線デジタル署名アルゴリズム (FIPS 186-2)。
BCRYPT_ECDSA_P521_ALGORITHM "ECDSA_P521"	521 ビット素数曲線デジタル署名アルゴリズム (FIPS 186-2)。

pszKeyName [in, optional]

キー名を含む null 終端された Unicode 文字列へのポインタ。このパラメータが NULL の場合、この関数は永続化されない一時キーを作成します。

dwLegacyKeySpec [in, unused]

AWS CloudHSM Key Storage Provider (KSP) はこのパラメータを使用しません。

dwFlags [in]

関数の動作を変更するためのフラグ。以下の値を 0 個以上使用します。

値	意味
NCRYPT_MACHINE_KEY_FLAG	このフラグは効果がありません。
NCRYPT_SILENT_FLAG	このフラグは効果がありません。
NCRYPT_OVERWRITE_KEY_FLAG	このフラグを指定すると、HSM 内の同じ名前の既存のキーが上書きされます。 このフラグがない場合、関数は NTE_EXISTS を返します。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	関数は正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。
NTE_NOT_SUPPORTED	pszAlgId パラメータにサポートされていない値が含まれています。
NTE_EXISTS	指定された名前のキーは既に存在し、オペレーションは NCRYPT_OVERWRITE_KEY_FLAG を使用しませんでした。

キーストレージプロバイダー (KSP) を使用した NCryptGetProperty

NCryptGetProperty 関数は、キーストレージオブジェクトのプロパティ値を取得します。

パラメータ

`hObject` [in]

プロパティを取得する対象のオブジェクトのハンドル。次を使用できます。

- プロバイダーハンドル (`NCRYPT_PROV_HANDLE`)
- キーハンドル (`NCRYPT_KEY_HANDLE`)

`pszProperty` [in]

取得するプロパティ名を含む null 終端 Unicode 文字列へのポインタ。

を使用する場合`NCRYPT_PROV_HANDLE`、 AWS CloudHSM キーストレージプロバイダー (KSP) は次の KSP 識別子をサポートします。

識別子/値	説明
NCRYPT_IMPL_TYPE_PROPERTY L"Impl Type"	プロバイダーの実装の詳細を定義するフラグを含む DWORD
NCRYPT_MAX_NAME_LENGTH_PROPERTY L"Max Name Length"	永続キー名の最大長 (文字数) を含む DWORD。
NCRYPT_NAME_PROPERTY L"Name"	KSP 名を含む、null 終端された Unicode 文字列へのポインタ。
NCRYPT_VERSION_PROPERTY L"Version"	プロバイダーのバージョンを含む DWORD (上位ワード: メジャーバージョン、下位ワード: マイナーバージョン)。
NCRYPT_USE_CONTEXT_PROPERTY L"Use Context"	オペレーションコンテキストを示す、null 終端された Unicode 文字列へのポインタ。
NCRYPT_SECURITY_DESCR_SUPPORT_PROPERTY	プロバイダーがキーに対するセキュリティ記述子をサポートしているかどうかを示します。

識別子/値	説明
L"Security Descr Support"	

を使用する場合NCRYPT_KEY_HANDLE、 AWS CloudHSM キーストレージプロバイダー (KSP) は次の KSP 識別子をサポートします。

識別子/値	説明
NCRYPT_ALGORITHM_PROPERTY L"Algorithm Name"	キーのアルゴリズム名を含む、null 終端された Unicode 文字列。
NCRYPT_BLOCK_LENGTH_PROPERTY L"Block Length"	暗号化ブロック長 (バイト単位) を含む DWORD。
NCRYPT_EXPORT_POLICY_PROPERTY L"Export Policy"	永続化されたキーのエクスポートポリシーを指定するフラグを含む DWORD。
NCRYPT_KEY_USAGE_PROPERTY L"Key Usage"	キー使用に関する詳細を定義するフラグを含む DWORD。
NCRYPT_KEY_TYPE_PROPERTY L"Key Type"	キータイプを定義するフラグを含む DWORD。
NCRYPT_LENGTH_PROPERTY L"Length"	キー長 (ビット数) を含む DWORD。
NCRYPT_LENGTHS_PROPERTY L"Lengths"	サポートされているキーサイズを含む NCRYPT_SUPPORTED_LENGTHS 構造体へのポインタ。
NCRYPT_NAME_PROPERTY L"Name"	キー名を含む、null 終端された Unicode 文字列へのポインタ。

識別子/値	説明
NCRYPT_SECURITY_DESCR_PROPERTY L"Security Descr"	キーのアクセス制御情報を含む SECURITY_DESCRIPTOR 構造体へのポインタ。
NCRYPT_ALGORITHM_GROUP_PROPERTY L"Algorithm Group"	オブザーバーのアルゴリズムグループ名を含む、null 終端された Unicode 文字列。
NCRYPT_UNIQUE_NAME_PROPERTY L"Unique Name"	キーの一意名を含む、null 終端された Unicode 文字列へのポインタ。

> **Note**
>
> 値は、リテラルの前に L が付いていることからわかるように、ワイド文字列リテラルです。

`pbOutput` [out]

プロパティ値を保存するバッファのアドレス。`cbOutput` を使用してバッファサイズを指定します。

必要なバッファサイズを決定するには、このパラメータを NULL に設定します。この関数は、必要なサイズ (バイト単位) を `pcbResult` が指す場所に保存します。

`cbOutput` [in]

`pbOutput` バッファのサイズ (バイト単位)。

`pcbResult` [out]

`pbOutput` バッファにコピーされたバイト数を保存する DWORD 変数へのポインタ。

`pbOutput` が NULL の場合、必要なサイズ (バイト単位) が保存されます。

`dwFlags` [in]

関数の動作を変更するためのフラグ。ゼロまたは以下の値を使用できます。

値	意味
NCRYPT_SILENT_FLAG	このフラグは効果がありません。

pszProperty が NCRYPT_SECURITY_DESCR_PROPERTY の場合、次のいずれかまたは組み合わせを使用します。

値	意味
OWNER_SECURITY_INFORMATION	このフラグは効果がありません。
GROUP_SECURITY_INFORMATION	このフラグは効果がありません。
DACL_SECURITY_INFORMATION	このフラグは効果がありません。
LABEL_SECURITY_INFORMATION	このフラグは効果がありません。
SACL_SECURITY_INFORMATION	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。
NTE_NOT_SUPPORTED	pszAlgId パラメータには、サポートされていない値が含まれています。

リターンコード	説明
NTE_INVALID_HANDLE	hObject のハンドルが無効です。
NTE_BUFFER_TOO_SMALL	cbOutput パラメータが、戻り値を格納するには小さすぎます。

キーストレージプロバイダー (KSP) を使用した NCryptSetProperty

NCryptSetProperty 関数は、キーストレージオブジェクトのプロパティ値を設定します。

パラメータ

hObject [in]

プロパティを設定する対象のオブジェクトのハンドル。次を使用できます。

- プロバイダーハンドル (NCRYPT_PROV_HANDLE)
- キーハンドル (NCRYPT_KEY_HANDLE)

pszProperty [in]

取得するプロパティ名を含む null 終端 Unicode 文字列へのポインタ。

を使用する場合NCRYPT_PROV_HANDLE、 AWS CloudHSM キーストレージプロバイダー (KSP) は次の KSP 識別子をサポートします。

識別子/値	説明
NCRYPT_USE_CONTEXT_PROPERTY L"Use Context"	オペレーションコンテキストを示す、null 終端された Unicode 文字列へのポインタ。

を使用する場合NCRYPT_KEY_HANDLE、 AWS CloudHSM キーストレージプロバイダー (KSP) は次の KSP 識別子をサポートします。

識別子/値	説明
NCRYPT_KEY_USAGE_PROPERTY	キー使用に関する詳細を定義するフラグのセットを含む DWORD。このプロパティは、

識別子/値	説明
L"Key Usage"	キーにのみ適用されます。これは、0 個、または以下の値の 1 つ以上の組み合わせを含めることができます。 NCRYPT_ALLOW_DECRYPT_FLAG (0x00000001) NCRYPT_ALLOW_SIGNING_FLAG (0x00000002)
NCRYPT_LENGTH_PROPERTY L"Length"	キー長 (ビット数) を含む DWORD。
NCRYPT_EXPORT_POLICY_PROPERTY L"Export Policy"	永続化されたキーのエクスポートポリシーを指定するフラグを含む DWORD。これは、0 個、または以下の値の 1 つ以上の組み合わせを含めることができます。 NCRYPT_ALLOW_EXPORT_FLAG (0x00000001)

> **Note**
>
> 値は、リテラルの前に L が付いていることからわかるように、ワイド文字列リテラルです。

`pbInput` [in]

新しいプロパティ値を含むバッファのアドレス。`cbInput` にはバッファのサイズが含まれます。

`cbInput` [in]

`pbInput` バッファのサイズ (バイト単位)。

`dwFlags` [in]

関数の動作を変更するフラグ。この関数にはフラグが定義されていません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。
NTE_NOT_SUPPORTED	pszProperty パラメータには、サポートされていない値が含まれています。
NTE_INVALID_HANDLE	hObject のハンドルが無効です。
NTE_BAD_DATA	pbInput および cbInput が指すデータは有効ではありません。

キーストレージプロバイダー (KSP) を使用した NCryptFinalizeKey

NCryptFinalizeKey 関数は KSP キーを完了します。キーを使用する前に、この関数を呼び出す必要があります。

パラメータ

hKey [in]

完了するキーのハンドル。NCryptCreatePersistedKey 関数を呼び出して、このハンドルを取得します。

dwFlags [in]

関数の動作を変更するためのフラグ。0 または次の値を使用できます。

値	意味
NCRYPT_SILENT_FLAG	このフラグは効果がありません。
NCRYPT_NO_KEY_VALIDATION	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_INVALID_HANDLE	hKey のハンドルが無効です。
NTE_NOT_SUPPORTED	dwFlags パラメータには、サポートされていない値が含まれています。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。

キーストレージプロバイダー (KSP) を使用した NCryptDeleteKey

NCryptDeleteKey 関数は、キーストレージプロバイダー (KSP) から KSP キーを削除します。

パラメータ

hKey [in]

削除するキーのハンドル。

dwFlags [in]

関数の動作を変更するためのフラグ。以下の値を 0 個以上使用できます。

値	意味
NCRYPT_SILENT_FLAG	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	関数は成功しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_INVALID_HANDLE	hKey のハンドルが無効です。
NTE_INTERNAL_ERROR	キーの削除中に内部エラーが発生しました。

キーストレージプロバイダー (KSP) を使用した NCryptFreeObject

NCryptFreeObject 関数は、キーストレージプロバイダー (KSP) からプロバイダーまたはキーハンドルを解放します。

パラメータ

hObject [in]

解放するオブジェクトのハンドル。次を使用できます。

- プロバイダーハンドル (NCRYPT_PROV_HANDLE)
- キーハンドル (NCRYPT_KEY_HANDLE)

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_HANDLE	`hObject` のハンドルが無効です。

キーストレージプロバイダー (KSP) を使用した NCryptFreeBuffer

`NCryptFreeBuffer` 関数は、キーストレージプロバイダー (KSP) によって割り当てられたメモリのブロックを解放します。

パラメータ

`pvInput` [in]

解放するメモリのアドレス。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_FAIL	オペレーションを完了できませんでした。

キーストレージプロバイダー (KSP) を使用した NCryptIsAlgSupported

NCryptIsAlgSupported 関数は、キーストレージプロバイダー (KSP) が特定の暗号アルゴリズムをサポートしているかどうかを判定します。

パラメータ

`hProvider` [in]

キーストレージプロバイダーのハンドル。このハンドルは、`NCryptOpenStorageProvider` を使用して取得します。

`pszAlgId` [in]

キーを作成するための暗号アルゴリズムの識別子を含む、null 終端された Unicode 文字列へのポインタ。AWS CloudHSM キーストレージプロバイダー (KSP) がサポートするアルゴリズムは以下のとおりです。

定数/値。	説明
BCRYPT_RSA_ALGORITHM "RSA"	RSA パブリックキーアルゴリズム。
BCRYPT_ECDSA_P256_ALGORITHM "ECDSA_P256"	256 ビット素数曲線デジタル署名アルゴリズム (FIPS 186-2)。
BCRYPT_ECDSA_P384_ALGORITHM "ECDSA_P384"	384 ビット素数曲線デジタル署名アルゴリズム (FIPS 186-2)。
BCRYPT_ECDSA_P521_ALGORITHM "ECDSA_P521"	521 ビット素数曲線デジタル署名アルゴリズム (FIPS 186-2)。

`dwFlags` [in]

関数の動作を変更するフラグ。これには、ゼロまたは次のいずれかの値を指定できます。

値	意味
NCRYPT_SILENT_FLAG	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。
NTE_NOT_SUPPORTED	pszAlgId パラメータにサポートされていない値が含まれています。
NTE_INVALID_HANDLE	hProvider のハンドルが無効です。

キーストレージプロバイダー (KSP) を使用した NCryptEnumAlgorithms

NCryptEnumAlgorithms 関数は、キーストレージプロバイダー (KSP) がサポートするアルゴリズムの名前を取得します。

パラメータ

hProvider [in]

アルゴリズムを列挙するキーストレージプロバイダーのハンドル。このハンドルを取得するには、NCryptOpenStorageProvider 関数を使用します。

dwAlgOperations [in]

列挙するアルゴリズムクラスを指定する値のセット。ゼロを使用してすべてのアルゴリズムを列挙することも、1 つ以上の値を結合することもできます。

値	意味
NCRYPT_ASYMMETRIC_ENCRYPTION_OPERATION	非対称暗号化アルゴリズムを一覧表示します。

値	意味
0x00000004	
NCRYPT_SIGNATURE_OPERATION 0x00000010	デジタル署名アルゴリズムを一覧表示します。

pdwAlgCount [out]

ppAlgList 配列内の要素の数を保存する DWORD のアドレス。

ppAlgList [out]

登録されたアルゴリズム名の配列を保存する NCryptAlgorithmName 構造ポインタのアドレス。pdwAlgCount パラメータは、この配列内の要素の数を示します。

dwFlags [in]

関数の動作を変更するためのフラグ。ゼロまたは次の値を使用します。

値	意味
NCRYPT_SILENT_FLAG	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。

リターンコード	説明
NTE_NOT_SUPPORTED	dwAlgOperations パラメータにサポートされていない値が含まれています。

キーストレージプロバイダー (KSP) を使用した NCryptEnumKeys

NCryptEnumKeys 関数は、キーストレージプロバイダー (KSP) に保存されているキーを一覧表示します。

パラメータ

hProvider [in]

キーストレージプロバイダーハンドル。このハンドルは、NCryptOpenStorageProvider を使用して取得します。

pszScope [in, unused]

このパラメータは NULL に設定します。

ppKeyName [out]

キー名を保存する NCryptKeyName 構造へのポインタアドレス。使用後にこのメモリを解放するには、NCryptFreeBuffer を呼び出します。

ppEnumState [in, out]

列挙の進行状況を追跡する VOID ポインタアドレス。キーストレージプロバイダーは、この情報を内部的に使用して列挙シーケンスを管理します。新しい列挙を最初から開始するには、このポインタを NULL に設定します。

列挙の完了後にこのメモリを解放するには、このポインタを NCryptFreeBuffer に渡します。

dwFlags [in]

関数の動作を変更するためのフラグ。この関数にはフラグはありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_INVALID_HANDLE	`hProvider` のハンドルが無効です。
NTE_NO_MORE_ITEMS	列挙には、使用可能なすべてのキーが一覧表示されています。

キーストレージプロバイダー (KSP) を使用した NCryptExportKey

`NCryptExportKey` 関数は KSP キーをメモリ BLOB にエクスポートします。この関数は、パブリックキーのエクスポートのみをサポートします。

パラメータ

`hKey` [in]

エクスポートするキーのハンドル。

`hExportKey` [in, unused]

AWS CloudHSM Key Storage Provider (KSP) はこのパラメータを使用しません。

`pszBlobType` [in]

エクスポートするBLOBタイプを指定する NULL 終了 Unicode 文字列。 AWS CloudHSM キーストレージプロバイダー (KSP) は次の値をサポートします。

値	意味
BCRYPT_RSAPUBLIC_BLOB	RSA パブリックキーをエクスポートします。`pbOutput` バッファには、キーデータの後に続く `BCRYPT_RSAKEY_BLOB` 構造が含まれています。

値	意味
BCRYPT_ECCPUBLIC_BLOB	ECC パブリックキーをエクスポートします。pbOutput バッファには、キーデータの後に続く BCRYPT_ECCKEY_BLOB 構造が含まれています。

pParameterList [in, unused]

AWS CloudHSM Key Storage Provider (KSP) はこのパラメータを使用しません。

pbOutput [out, optional]

キー BLOB を保存するバッファアドレス。cbOutput を使用してバッファサイズを指定します。NULL に設定すると、関数は pcbResult が指す DWORD に必要なサイズ (バイト単位) を保存します。

cbOutput [in]

pbOutput バッファのサイズ (バイト単位)。

pcbResult [out]

pbOutput バッファにコピーされたバイト数を保存する DWORD 変数アドレス。pbOutput が NULL の場合、関数は必要なバッファサイズをバイト単位で保存します。

dwFlags [in]

関数の動作を変更するためのフラグ。ゼロまたは以下の値を使用できます。

値	意味
NCRYPT_SILENT_FLAG	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_INVALID_HANDLE	`hProvider` のハンドルが無効です。
NTE_BAD_FLAGS	`dwFlags` パラメータに無効な値が含まれています。
NTE_BAD_KEY_STATE	キーの状態が無効です。
NTE_NOT_SUPPORTED	`pszBlobType` または `dwFlags` パラメータにサポートされていない値が含まれています。
STATUS_INTERNAL_ERROR	オペレーション中に内部エラーが発生しました。

キーストレージプロバイダー (KSP) を使用した NCryptSignHash

`NCryptSignHash` 関数はハッシュ値の署名を作成します。

パラメータ

`hKey` [in]

ハッシュの署名に使用するキーのハンドル。

`pPaddingInfo` [in, optional]

パディング情報を含む構造へのポインタ。構造タイプは `dwFlags` 値によって異なります。このパラメータは非対称キーでのみ使用し、その他のキータイプでは NULL に設定します。

`pbHashValue` [in]

署名するハッシュ値を含むバッファへのポインタ。`cbHashValue` を使用してバッファサイズを指定します。

cbHashValue [in]

署名する pbHashValue バッファのサイズ (バイト単位)。

pbSignature [out]

署名を保存するバッファのアドレス。cbSignature を使用してバッファサイズを指定します。

必要なバッファサイズを決定するには、このパラメータを NULL に設定します。この関数は、必要なサイズ (バイト単位) を pcbResult が指す場所に保存します。

cbSignature [in]

pbSignature バッファのサイズ (バイト単位)。この関数は、pbSignature が NULL の場合、このパラメータを無視します。

pcbResult [out]

pbSignature バッファにコピーされたバイト数を保存する DWORD 変数へのポインタ。

pbSignature が NULL の場合、必要なバッファサイズ (バイト単位) が保存されます。

dwFlags [in]

関数の動作を変更するためのフラグ。許可されるフラグは、キータイプによって異なります。次の値のいずれかを使用してください。

値	意味
BCRYPT_PAD_PKCS1	PKCS1 パディング方式を使用します。BCRYPT_PKCS1_PADDING_INFO 構造体を指すように pPaddingInfo を設定します。
BCRYPT_PAD_PSS	確率的署名方式 (PSS) パディング方式を使用します。BCRYPT_PSS_PADDING_INFO 構造体を指すように pPaddingInfo パラメータを設定します。
NCRYPT_SILENT_FLAG	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_INVALID_HANDLE	hKey のハンドルが無効です。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。
NTE_BUFFER_TOO_SMALL	pcbOutput パラメータが、戻り値を格納するには小さすぎます。
NTE_BAD_KEY_STATE	キーの状態が無効です。
NTE_INTERNAL_ERROR	ハッシュの署名時に内部エラーが発生しました。

キーストレージプロバイダー (KSP) を使用した NCryptVerifySignature

NCryptVerifySignature 関数は、署名が指定されたハッシュと一致するかどうかを確認します。

パラメータ

hKey [in]

署名を復号するために使用するキーのハンドル。NCryptSignHash でデータの署名に使用されたキーペアのパブリックキー部分を使用する必要があります。

pPaddingInfo [in, optional]

パディング情報を含む構造へのポインタ。構造タイプは dwFlags 値によって異なります。このパラメータは非対称キーでのみ使用し、その他のキータイプでは NULL に設定します。

pbHashValue [in]

署名するハッシュ値を含むバッファへのポインタ。cbHashValue を使用してバッファサイズを指定します。

cbHashValue [in]

pbHashValue バッファのサイズ (バイト単位)。

pbSignature [out]

データの署名付きハッシュを含むバッファのアドレス。この署名を作成するには NCryptSignHash を使用します。cbSignature を使用してバッファサイズを指定します。

cbSignature [in]

pbSignature バッファのサイズ (バイト単位)。この署名を作成するには NCryptSignHash を使用します。

dwFlags [in]

関数の動作を変更するためのフラグ。許可されるフラグは、キータイプによって異なります。次の値のいずれかを使用してください。

値	意味
NCRYPT_PAD_PKCS1_FLAG	署名に PKCS1 パディング が使用されたことを示します。BCRYPT_PKCS1_PADDING_INFO 構造体を指すように pPaddingInfo を設定します。
NCRYPT_PAD_PSS_FLAG	署名に確率的署名方式 (PSS) パディングが使用されたことを示します。BCRYPT_PSS_PADDING_INFO 構造体を指すように pPaddingInfo を設定します。
NCRYPT_SILENT_FLAG	このフラグは効果がありません。

戻り値

この関数は、成功または失敗を示すステータスコードを返します。

一般的なリターンコードは次のとおりです。

リターンコード	説明
ERROR_SUCCESS	オペレーションが正常に完了しました。
NTE_INVALID_PARAMETER	1 つ以上のパラメータが無効です。
NTE_FAIL	オペレーションを完了できませんでした。
NTE_INVALID_HANDLE	hKey のハンドルが無効です。
NTE_BAD_FLAGS	dwFlags パラメータに無効な値が含まれています。
NTE_BAD_SIGNATURE	署名は検証されませんでした。
NTE_BAD_KEY_STATE	キーの状態が無効です。
NTE_INTERNAL_ERROR	署名の検証中に内部エラーが発生しました。

の KSP の詳細設定 AWS CloudHSM

AWS CloudHSM キーストレージプロバイダー (KSP) には、次の高度な設定が含まれています。これは、ほとんどのお客様が使用する一般的な設定の一部ではありません。これらの設定には追加機能があります。

- KSP の SDK3 互換モード

のキーストレージプロバイダー (KSP) の SDK3 互換性モード AWS CloudHSM

キーストレージプロバイダー (KSP) は、HSM キーとの連携に対して異なるアプローチを実装しています。

- クライアント SDK 5: HSM に保存されたキーと直接通信し、ローカルの参照ファイルが不要
- クライアント SDK 3: Windows サーバー上に、HSM に保存されたキーを参照するローカルファイルを保持し、これらのファイルを使用してキーに関するオペレーションを実行

クライアント SDK 3 からクライアント SDK 5 へ移行するお客様向けに、SDK3 互換モードを有効にすると、HSM のキー保管アーキテクチャを保持したまま、既存のキー参照ファイルを使用したオペレーションをサポートできます。

SDK3 互換モードを有効にする

Windows

Windows でクライアント SDK 5 用キーストレージプロバイダー (KSP) の SDK3 互換モードを有効にするには

- 次のコマンドを使用して SDK3 互換モードを有効にできます。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --enable-sdk3-compatibility-mode
```

SDK3 互換モードを無効にする

Windows

Windows でクライアント SDK 5 用キーストレージプロバイダー (KSP) の SDK3 互換モードを無効にするには

- 次のコマンドを使用して SDK3 互換モードを無効にできます。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-sdk3-compatibility-mode
```

AWS CloudHSM クライアント SDK 5 の JCE プロバイダー

AWS CloudHSM JCE プロバイダーは、Java 暗号化拡張 (JCE) プロバイダーフレームワークから構築されたプロバイダー実装です。JCE では、Java 開発キット (JDK) を使用して暗号化操作を実行できます。このガイドでは、 AWS CloudHSM JCE プロバイダーは JCE プロバイダーと呼ばれることがあります。JCE プロバイダーと JDK を使用して、HSM に暗号化操作をオフロードします。トラブルシューティングについては、「JCE SDK for の既知の問題 AWS CloudHSM」を参照してください。

クライアント SDK 3 の使用の詳細については、「以前の SDK バージョンを使用して を操作する AWS CloudHSM」を参照してください。

トピック

- AWS CloudHSM クライアント SDK 5 の JCE プロバイダーをインストールする
- AWS CloudHSM クライアント SDK 5 の JCE プロバイダーでサポートされているキータイプ
- AWS CloudHSM クライアント SDK 5 の JCE プロバイダーにおけるキー管理の基本
- AWS CloudHSM クライアント SDK 5 の JCE プロバイダーでサポートされているメカニズム
- AWS CloudHSM クライアント SDK 5 でサポートされている Java キー属性
- Java for Client SDK 5 用の AWS CloudHSM ソフトウェアライブラリのコードサンプル
- AWS CloudHSM JCE プロバイダー Javadocs
- クライアント SDK 5 のAWS CloudHSM KeyStore Java クラス
- AWS CloudHSM JCE for Client SDK 5 の詳細設定

AWS CloudHSM クライアント SDK 5 の JCE プロバイダーをインストールする

AWS CloudHSM クライアント SDK 5 の JCE プロバイダーは、OpenJDK 17、OpenJDK 21、および OpenJDK 25 と互換性があります。OpenJDK ウェブサイトから OpenJDK をダウンロードできます。

> **Note**
>
> - クライアント SDK 5.17.0 は、OpenJDK 8 をサポートする最後のリリースです。
> - クライアント SDK 5.17.1 は、OpenJDK 11 をサポートする最後のリリースです。

以下のセクションを使用して、プロバイダーをインストールし、認証情報をプロバイダーに提供します。

> **Note**
>
> クライアント SDK 5 で単一の HSM クラスターを実行するには、まず `disable_key_availability_check` を `True` に設定してクライアントキーの耐久性の設定を管理する必要があります。詳細については、キーの同期 と クライアント SDK 5 設定ツール を参照してください。

トピック

- ステップ 1: JCE プロバイダーをインストールする
- ステップ 2: JCE プロバイダーに認証情報を提供する

ステップ 1: JCE プロバイダーをインストールする

1. 以下のコマンドを使用して、JCE プロバイダーをダウンロードし,インストールします。

   Amazon Linux 2023

   X86_64 アーキテクチャ上で Amazon Linux 2023 用の JCE プロバイダーをインストールします。

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-jce-latest.amzn2023.x86_64.rpm
   ```

   Amazon Linux 2023 用の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.aarch64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-jce-latest.amzn2023.aarch64.rpm
   ```

   Amazon Linux 2

   X86_64 アーキテクチャ上で Amazon Linux 2 用の JCE プロバイダーをインストールします。

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
   ```

   Amazon Linux 2 用の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

RHEL 10 の JCE プロバイダーを x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-jce-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el10.x86_64.rpm
```

RHEL 10 の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-jce-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

RHEL 9 (9.2 以降) 用の JCE プロバイダーを x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.x86_64.rpm
```

RHEL 9 (9.2 以降) 用の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

RHEL 8 用の JCE プロバイダーを x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el8.x86_64.rpm
```

RHEL 8 の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-jce-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el8.aarch64.rpm
```

Ubuntu 26.04 LTS

Ubuntu 26.04 LTS 用の JCE プロバイダーを x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/cloudhsm-jce_latest_u26.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u26.04_amd64.deb
```

Ubuntu 26.04 LTS 用の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Resolute/cloudhsm-jce_latest_u26.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u26.04_arm64.deb
```

Ubuntu 24.04 LTS

Ubuntu 24.04 LTS 用の JCE プロバイダーを x86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/
cloudhsm-jce_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u24.04_amd64.deb
```

Ubuntu 24.04 LTS 用の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/
cloudhsm-jce_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS 用の JCE プロバイダーを X86_64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-jce_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_amd64.deb
```

Ubuntu 22.04 LTS 用の JCE プロバイダーを ARM64 アーキテクチャにインストールします。

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-jce_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_arm64.deb
```

Windows Server

Windows Server 用の JCE プロバイダーを x86_64 アーキテクチャにインストールし、管理者として PowerShell を開き、以下のコマンドを実行します。

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/
Windows/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMJCE-
latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

2. クライアント SDK 5 をブートストラップします。ブートストラップの詳細については、クライアント SDK をブートストラップする を参照してください。
3. 次の JCE プロバイダーファイルを見つけます。

   Linux

   - `/opt/cloudhsm/java/cloudhsm-<version>.jar`
   - `/opt/cloudhsm/bin/configure-jce`
   - `/opt/cloudhsm/bin/jce-info`

   Windows

   - `C:\Program Files\Amazon\CloudHSM\java\cloudhsm-<version>.jar>`
   - `C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe`
   - `C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe`

ステップ 2: JCE プロバイダーに認証情報を提供する

HSM では、Java アプリケーションが HSM を使用する前に、Java アプリケーションを認証する必要があります。HSM は、明示的なログインと暗黙的なログイン方法のいずれかを使用して認証します。

明示的なログイン - この方法では、 AWS CloudHSM 認証情報をアプリケーションに直接渡すことができます。CU ユーザー名とパスワードを PIN パターンで渡す `AuthProvider` を使用します。詳細については、「Login to an HSM」のコード例を参照してください。

暗黙的なログイン - この方法では、 AWS CloudHSM 認証情報を、新しいプロパティファイルまたはシステムプロパティで設定するか、環境変数として設定することができます。

- System properties - アプリケーションの実行時に、システムプロパティを通して認証情報を設定します。次の例は、これを行うための 2 つの異なる方法を示しています。

  Linux

  ```
  $ java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
  ```

  ```
  System.setProperty("HSM_USER","<HSM user name>");
  System.setProperty("HSM_PASSWORD","<password>");
  ```

  Windows

  ```
  PS C:\> java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
  ```

  ```
  System.setProperty("HSM_USER","<HSM user name>");
  System.setProperty("HSM_PASSWORD","<password>");
  ```

- Environment variables - 認証情報を環境変数として設定します。

  > **Note**
  >
  > 環境変数を設定するときは、シェルによって解釈される可能性のある特殊文字を エスケープする必要があります。

  Linux

  ```
  $ export HSM_USER=<HSM user name>
  $ export HSM_PASSWORD=<password>
  ```

  Windows

  ```
  PS C:\> $Env:HSM_USER="<HSM user name>"
  PS C:\> $Env:HSM_PASSWORD="<password>"
  ```

アプリケーションで設定されない場合、または HSM でセッションを認証する前にユーザーが操作を行った場合は、認証情報を使用できない場合があります。このような場合は、Java 用の CloudHSM ソフトウェアライブラリによって、次の順序で認証情報が検索されます。

1. システムプロパティ
2. 環境変数

AWS CloudHSM クライアント SDK 5 の JCE プロバイダーでサポートされているキータイプ

Java 用の AWS CloudHSM ソフトウェアライブラリでは、次のキータイプを生成できます。

キータイプ	説明
AES	128、192、256 ビットの AES キーを生成します。
Triple DES (3DES、DESede)	192 ビットトリプル DES キー を生成します[1]。
EC	EC キーペアの生成 – NIST 曲線 secp224r1 (P-224)、secp256r1 (P-256)、secp256k1 (ブロックチェーン)、secp384r1 (P-384)、secp521r1 (P-521)、ed25519[2]。
[GENERIC_SECRET]	1 ~ 800 バイトの汎用シークレットを生成します。
HMAC	SHA1、SHA224、SHA256、SHA384、SHA512 のハッシュサポート。
ML-DSA	パラメータセット ML-DSA[3]-ML-DSA-44ML-DSA-65ML-DSA-87キーペアを生成します。
RSA	256 ビットの増分で、2,048 ~ 4,096 ビットの RSA キーを生成します。

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

[2] 非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

[3] 2026 年 9 月 1 日以降、ML-DSA は hsm2m.medium クラスターの FIPS モードで使用できます。

AWS CloudHSM クライアント SDK 5 の JCE プロバイダーにおけるキー管理の基本

JCE プロバイダー中のキー管理の基本には、キーのインポート、キーのエクスポート、ハンドルによるキーのロード、またはキーの削除などがあります。キーの管理の詳細については、「キーの管理」のサンプルコードを参照してください。

また、JCE プロバイダーのサンプルコードについては、コードサンプル で参照できます。

AWS CloudHSM クライアント SDK 5 の JCE プロバイダーでサポートされているメカニズム

このトピックでは、 AWS CloudHSM クライアント SDK 5 で JCE プロバイダーでサポートされているメカニズムについて説明します。でサポートされている Java 暗号化アーキテクチャ (JCA) インターフェイスとエンジンクラスの詳細については AWS CloudHSM、以下のトピックを参照してください。

トピック

- キーとキーペアの関数を生成する
- 暗号関数
- 署名および検証
- ダイジェスト関数
- Hash-based Message Authentication Code (HMAC) 関数
- 暗号ベースのメッセージ認証コード (CMAC) 関数
- キー共有機能
- キーファクトリを使用してキーをキー仕様に変換します
- メカニズムの注釈

キーとキーペアの関数を生成する

Java 用 AWS CloudHSM ソフトウェアライブラリでは、次のオペレーションを使用してキーとキーペア関数を生成できます。

- RSA
- EC
- EdDSA (Ed25519)[注記「2」を参照]
- ML-DSA
- AES
- DESede (Triple DES)[注記「1」を参照]
- GenericSecret

暗号関数

Java 用 AWS CloudHSM ソフトウェアライブラリは、次のアルゴリズム、モード、パディングの組み合わせをサポートしています。

アルゴリズム	モード	[Padding] (パディング)	注意事項
AES	CBC	AES/CBC/NoPadding AES/CBC/PKCS5Padding	Cipher.ENCRYPT_MODE および Cipher.DECRYPT_MODE を実装します。 Cipher.UNWRAP_MODE for AES/CBC NoPadding を実装します
AES	ECB	AES/ECB/PKCS5Padding	Cipher.ENCRYPT_MODE および Cipher.DE

アルゴリズム	モード	[Padding] (パディング)	注意事項
		AES/ECB/NoPadding	CRYPT_MODE を実装します。
AES	CTR	AES/CTR/NoPadding	Cipher.ENCRYPT_MODE および Cipher.DECRYPT_MODE を実装します。
AES	GCM	AES/GCM/NoPadding	Cipher.WRAP_MODE 、Cipher.UNWRAP_MODE 、Cipher.ENCRYPT_MODE 、および Cipher.DECRYPT_MODE を実装します。 AES-GCM 暗号化の実行時に、HSM はリクエスト内の初期化ベクトル (IV) を無視し、独自に IV を生成して使用します。オペレーションが完了したら、Cipher.getIV() を呼び出して IV を取得する必要があります。

アルゴリズム	モード	[Padding] (パディング)	注意事項

AESWrap	ECB	`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
		
		`AESWrap/ECB/ZeroPadding`
			`Cipher.WRAP_MODE` および `Cipher.UNWRAP_MODE` を実装します。

DESede (Triple DES)	CBC	`DESede/CBC/PKCS5Padding`
		
		`DESede/CBC/NoPadding`
			`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。今後の変更については、以下の注記「1」を参照してください。

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。今後の変更については、以下の注記「1」を参照してください。

アルゴリズム	モード	[Padding] (パディング)	注意事項
RSA	ECB	RSA/ECB/PKCS1Padding 注記「1」を参照 RSA/ECB/OAEPPadding RSA/ECB/OAEPWithSHA-1ANDMGF1Padding RSA/ECB/OAEPWithSHA-224ANDMGF1Padding RSA/ECB/OAEPWithSHA-256ANDMGF1Padding RSA/ECB/OAEPWithSHA-384ANDMGF1Padding RSA/ECB/OAEPWithSHA-512ANDMGF1Padding	Cipher.WRAP_MODE 、Cipher.UNWRAP_MODE 、Cipher.ENCRYPT_MODE 、および Cipher.DECRYPT_MODE を実装します。

アルゴリズム	モード	[Padding] (パディング)	注意事項
RSA	ECB	RSA/ECB/NoPadding	Cipher.ENCRYPT_MODE および Cipher.DECRYPT_MODE を実装します。
RSAAESWrap	ECB	RSAAESWrap/ECB/OAEPPadding RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding	Cipher.WRAP_MODE および Cipher.UNWRAP_MODE を実装します。

署名および検証

Java 用 AWS CloudHSM ソフトウェアライブラリは、次のタイプの署名と検証をサポートしています。クライアント SDK 5 とハッシュ機能付きの署名アルゴリズムでは、データはソフトウェアでローカルにハッシュされてから、署名/検証のために HSM に送信されます。つまり、SDK でハッシュできるデータのサイズに制限はありません。

RSA 署名タイプ

- NONEwithRSA
- RSASSA-PSS
- SHA1withRSA
- SHA1withRSA/PSS
- SHA1withRSAandMGF1
- SHA224withRSA
- SHA224withRSAandMGF1
- SHA224withRSA/PSS
- SHA256withRSA
- SHA256withRSAandMGF1
- SHA256withRSA/PSS
- SHA384withRSA
- SHA384withRSAandMGF1
- SHA384withRSA/PSS
- SHA512withRSA
- SHA512withRSAandMGF1
- SHA512withRSA/PSS

ECDSA 署名タイプ

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA

- SHA512withECDSA

ML-DSA 署名タイプ

- ML-DSA
- ML-DSA-44
- ML-DSA-65
- ML-DSA-87
- ML-DSA-EXTERNAL-MU

EdDSA 署名タイプ [注を参照 2]

- Ed25519 または EdDSA (最大データサイズ: 16,000 バイト)
- Ed25519ph

ダイジェスト関数

Java 用 AWS CloudHSM ソフトウェアライブラリは、次のメッセージダイジェストをサポートしています。クライアント SDK 5 では、データはソフトウェアでローカルにハッシュされます。つまり、SDK でハッシュできるデータのサイズに制限はありません。

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

Hash-based Message Authentication Code (HMAC) 関数

Java 用の AWS CloudHSM ソフトウェアライブラリは、次の HMAC アルゴリズムをサポートしています。

- HmacSHA1 (最大データサイズ (バイト): 16288)
- HmacSHA224 (最大データサイズ (バイト): 16256)
- HmacSHA256 (最大データサイズ (バイト): 16288)

- HmacSHA384 (最大データサイズ (バイト): 16224)
- HmacSHA512 (最大データサイズ (バイト): 16224)

暗号ベースのメッセージ認証コード (CMAC) 関数

CMAC (暗号ベースのメッセージ認証コード) は、ブロック暗号とシークレットキーを使用してメッセージ認証コード (MAC) を作成します。HMAC とは異なり、MAC にはハッシュ方式ではなくブロック対称キーメソッドを使用します。

Java 用 AWS CloudHSM ソフトウェアライブラリは、次の CMAC アルゴリズムをサポートしています。

- AESCMAC

キー共有機能

Java 用の AWS CloudHSM ソフトウェアライブラリは、Key Derivation Functions (KDF) を使用した ECDH をサポートしています。サポートされている KDF の種類は次のとおりです。

- ECDHwithX963SHA1KDF X9.63 KDF (SHA1 アルゴリズム) をサポート[3]
- ECDHwithX963SHA224KDF X9.63 KDF (SHA224 アルゴリズム) をサポート[3]
- ECDHwithX963SHA256KDF X9.63 KDF (SHA256 アルゴリズム) をサポート[3]
- ECDHwithX963SHA384KDF X9.63 KDF (SHA384 アルゴリズム) をサポート[3]
- ECDHwithX963SHA512KDF X9.63 KDF (SHA512 アルゴリズム) をサポート[3]

キーファクトリを使用してキーをキー仕様に変換します

キーファクトリを使用してキーをキー仕様に変換できます。 AWS CloudHSM には、JCE 用の 2 種類のキーファクトリがあります。

SecretKeyFactory: 対称キーをインポートまたは派生させるために使用されます。SecretKeyFactory を使用すると、サポートされているキーまたはサポートされている KeySpec を渡して、対称キーを AWS CloudHSMにインポートまたは派生させることができます。KeyFactory でサポートされている仕様は次のとおりです。

- SecretKeyFactory generateSecret のメソッドでは、以下の KeySpec クラスがサポートされています。

- KeyAttributesMap を使用して、追加の属性を持つキーバイトを CloudHSM キーとしてインポートできます。例はこちらからご覧いただけます。
- SecretKeySpec を使用して、対称キー仕様を CloudHSM キーとしてインポートできます。
- AesCmacKdfParameterSpec を使用して、別の CloudHSM AES キーを使用して対称キーを取得できます。

> **Note**
>
> SecretKeyFactory の `translateKey` メソッドは、キーインターフェイスを実装する任意のキーを受け取ります。

KeyFactory: 非対称キーのインポートに使用されます。SecretKeyFactory を使用すると、サポートされているキーまたはサポートされている KeySpec を渡して、非対称キーを AWS CloudHSMにインポートすることができます。詳細については、以下のリソースを参照してください。

- KeyFactory `generatePublic` のメソッドでは、次の KeySpec クラスがサポートされています。
  - RSA、EC、および EdDSA KeyTypes 用の CloudHSM KeyAttributesMap。 KeyAttributesMap KeyTypes 例はこちらからご覧いただけます
  - RSA、EC、および EdDSA パブリックキー用の X509EncodedKeySpec
  - RSA パブリックキー向け RSAPublicKeySpec
  - EC パブリックキーの ECPublicKeySpec (Ed25519 には適用されません)
- KeyFactory `generatePrivate` のメソッドでは、次の KeySpec クラスがサポートされています。
  - RSA、EC、および EdDSA KeyTypes 用の CloudHSM KeyAttributesMap。 KeyAttributesMap KeyTypes 例はこちらからご覧いただけます
  - RSA、EC、および EdDSA プライベートキーの PKCS8EncodedKeySpec
  - RSA プライベートキー向け RSAPrivateCrtKeySpec
  - EC プライベートキーの ECPrivateKeySpec (Ed25519 には適用されません)

> **ML-DSA キーのインポート制限**
>
> AWS CloudHSM は、 `KeyFactory` (`generatePublic` および) を介した ML-DSA キーのインポートをサポートしていませんgeneratePrivate。ML-DSA キーを使用するに

は、 を使用して HSM で直接生成しますKeyPairGenerator。ML-DSA は、 を使用したパブリックキーのエクスポートKeyFactoryをサポートします getKeySpec (を返しますX509EncodedKeySpec)。

KeyFactory の translateKey メソッドでは、キーインターフェイスを実装する任意のキーが取り込まれます。

メカニズムの注釈

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

[2] 非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

[3] キー取得関数 (KDFs) は NIST Special Publication 800-56A Revision 3 で指定されています。

AWS CloudHSM クライアント SDK 5 でサポートされている Java キー属性

このトピックでは、 AWS CloudHSM クライアント SDK 5 でサポートされている Java キー属性について説明します。このトピックでは、JCE プロバイダーの独自の拡張機能を使用してキーの属性を設定する方法について説明します。この拡張機能を使用して、これらのオペレーション中にサポートされるキー属性とその値を設定します。

- キー生成
- キーのインポート

キーアトリビュートの使用方法の例については、「the section called “コードサンプル”」を参照してください。

トピック

- 属性について
- サポートされている 属性
- キーの属性設定

属性について

キー属性を使用して、パブリックキー、プライベートキー、シークレットキーなど、キーオブジェクトで許可されるアクションを指定します。キー属性と値は、キーオブジェクトの作成オペレーション中に定義されます。

Java Cryptography Extension (JCE) では、キー属性に値を設定する方法が指定されていないため、ほとんどのアクションがデフォルトで許可されていました。これに対して、PKCS # 11 標準では、より制限の厳しいデフォルトのある包括的な属性のセットが定義されています。JCE プロバイダー 3.1 以降、 は一般的に使用される属性に対してより制限の厳しい値を設定できる独自の拡張機能 AWS CloudHSM を提供します。

サポートされている 属性

次の表に示す属性の値を設定できます。ベストプラクティスとして、制限する属性の値のみを設定してください。値を指定しない場合、 は以下の表で指定されたデフォルト値 AWS CloudHSM を使用します。デフォルト値の列のセルが空の場合は、属性に割り当てられている特定のデフォルト値がないことを示します。

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
DECRYPT	TRUE		TRUE[1]	True は、キーを使用して任意のバッファを復号できることを示します。通常、WRAP「true」に設定されているキーに対しては、これを「FALSE」に設定します。
DERIVE				キーを使用して他のキーを派生

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
				させることができます。
ENCRYPT	TRUE	TRUE[1]		True は、キーを使用して任意のバッファを暗号化できることを示します。
EXTRACTABLE	TRUE		TRUE	True は、このキーを HSM からエクスポートできることを示します。
ID				キーを識別するためにユーザーが定義する値。
KEY_TYPE				キーのタイプ (AES、DESede、汎用シークレット、EC、RSA、または ML-DSA) を識別するために使用されます。

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
LABEL				HSM 上のキーを簡単に識別できるユーザー定義の文字列。ベストプラクティスに従うには、後で見つけやすいように、各キーに固有のラベルを付けてください。
LOCAL				HSM によって生成されたキーを示します。
MLDSA_ALGORITHM				ML-DSA キーペアの生成に使用される ML-DSA パラメータセットを指定します。有効な値: ML-DSA-44、ML-DSA-65、および ML-DSA-87。

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
OBJECT_CLASS				キー (SecretKey、PublicKey、またはPrivateKey) のオブジェクトクラスを識別するために使用されます。
PRIVATE	TRUE	TRUE	TRUE	True は、ユーザーが認証されるまで、ユーザーがキーにアクセスできないことを示します。わかりやすくするために、この属性が FALSE に設定されている場合でも、ユーザーは認証 AWS CloudHSM されるまで のキーにアクセスできません。

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
SIGN	TRUE		TRUE	True は、キーを使用してメッセージダイジェストに署名できることを示します。パブリックキーおよびアーカイブしたプライベートキーの場合、通常 FALSE に設定されます。
SIZE				キーのサイズを定義する属性。サポートされているキーサイズの詳細については、「クライアント SDK 5 でサポートされているメカニズム」を参照してください。

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
TOKEN	FALSE	FALSE	FALSE	クラスター内のすべての HSM にレプリケートされ、バックアップに含まれる永続的なキー。TOKEN = FALSE は、HSM への接続が切断されるかログアウトされると自動的に消去される一時キーを意味します。
UNWRAP	TRUE		TRUE[1]	True は、キーを使用して別のキーをラップ解除 (インポート) できることを示します。
VERIFY	TRUE	TRUE		True は、キーを使用して署名を検証できることを示します。これは通常、プライベートキーの場合、FALSE に設定されます。

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
WRAP	TRUE	TRUE[1]		True は、キーを使用して別のキーをラップできることを示します。通常、プライベートキーの場合、これを FALSE に設定します。

属性	デフォルト値			注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	
WRAP_WITH_TRUSTED	FALSE		FALSE	True は、TRUSTED 属性が true に設定されているキーでのみキーのラップとラップ解除ができることを示します。キーが true に WRAP_WITH_TRUSTED 設定されると、その属性は読み取り専用になり、false に設定できなくなります。トラストラッピングについて詳しくは、「信頼できるキーを使ってキーのアンラップを制御する」をご覧ください。

[1] Ed25519 キーの場合、ENCRYPT、WRAP、DECRYPT、UNWRAP は機能しません。Ed25519 キーは署名専用であり、暗号化またはラッピングオペレーションをサポートしていません。Ed25519 キーをインポートする場合、これらの属性は FALSE に設定されます。

> **Note**
>
> PKCS #11 ライブラリでは、より広範な属性がサポートされます。詳細については、「サポートされている PKCS #11 属性」を参照してください。

キーの属性設定

`KeyAttributesMap` は Java Map のようなオブジェクトで、キーオブジェクトの属性値を設定するために使用できます。`KeyAttributesMap` 関数のメソッドは、Java マップ操作のメソッドと同様です。

属性にカスタム値を設定するには、次の 2 つのオプションがあります。

- 次の表に示す方法を使用します。
- このドキュメントの後半で説明するビルダーパターンの使用

属性マップオブジェクトは、属性を設定するための次のメソッドをサポートしています。

運用	戻り値	**`KeyAttributesMap`** 方法
既存のキーのキー属性の値を取得する	オブジェクト (値を含む) または null	get(keyAttribute)
1 つのキー属性の値を入力します。	キー属性のマッピングがなかった場合、キー属性に関連付けられた以前の値、または null	put(keyAttribute, value)
複数のキー属性の値を設定する	該当なし	putAll(keyAttributesMap)
属性マップからキーと値のペアを削除する	キー属性のマッピングがなかった場合、キー属性に関連付けられた以前の値、または null	remove(keyAttribute)

> **Note**
>
> 明示的に指定しない属性は、上記の the section called “サポートされている 属性” の表に示したデフォルトに設定されます。

キーペアの属性設定

Java クラス `KeyPairAttributesMap` を使用して、キーペアのキー属性を処理します。`KeyPairAttributesMap` は、2 つの `KeyAttributesMap` オブジェクトをカプセル化します。1 つはパブリックキー用ともう 1 つはプライベートキー用です。

パブリックキーとプライベートキーの個々の属性を個別に設定するには、そのキーの対応する `KeyAttributes` マップオブジェクトで `put()` メソッドを使用できます。`getPublic()` メソッドを使用してパブリックキーの属性マップを取得し、`getPrivate()` を使用してプライベートキーの属性マップを取得します。引数としてキーペア属性マップを使用する `putAll()` を使用して、パブリックキーペアとプライベートキーペアの両方に、複数のキー属性の値を一緒に入力します。

Java for Client SDK 5 用の AWS CloudHSM ソフトウェアライブラリのコードサンプル

このトピックでは、 AWS CloudHSM クライアント SDK 5 の Java コードサンプルに関するリソースと情報を提供します。

前提条件

サンプルを実行する前に、環境をセットアップする必要があります。

- Java Cryptographic Extension (JCE) provider をインストールして設定します。
- 有効な HSM ユーザー名とパスワードを設定します。これらのタスクには、暗号化ユーザー (CU) のアクセス権限で十分です。アプリケーションは、それぞれの例でこの認証情報を使用して HSM にログインします。
- JCE provider へのクレデンシャルを提供する方法を決定します。

コードサンプル

次のコードサンプルでは、基本タスクを実行するために、AWS CloudHSM JCE provider を使用する方法を示します。その他の例は GitHub から入手できます。

- HSM へのログイン
- キーの管理
- 対称キーの生成
- 非対称キーの生成
- AES GCM による暗号化と復号
- Encrypt and decrypt with AES-CTR
- DESede-ECB による暗号化と復号化 注記 1 参照
- Sign and Verify with RSA Keys
- Sign and Verify with EC Keys
- EdDSA キーによる署名と検証 注を参照 2
- ML-DSA キーによる署名と検証
- サポートされているキー属性の使用
- CloudHSM キーストアの使用

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

[2] 非 FIPS モードの hsm2m.medium インスタンスでのみサポートされています。

AWS CloudHSM JCE プロバイダー Javadocs

JCE プロバイダーの Javadocs を使用して、AWS CloudHSM JCE SDK で定義されている Java タイプとメソッドに関する使用情報を取得します。の最新の Javadocs をダウンロードするには AWS CloudHSM、ダウンロードページの AWS CloudHSM 最新の Client SDK リリースセクションを参照してください。

Javadocs は統合開発環境 (IDE) にインポートしたり、ウェブブラウザで表示することができます。

クライアント SDK 5 のAWS CloudHSM KeyStore Java クラス

クラスは AWS CloudHSM `KeyStore`、専用 PKCS12 キーストアを提供します。このキーストアでは、証明書をキーデータとともに保存し、 AWS CloudHSMに保存されているキーデータに関連付けることができます。`KeyStore` クラスは、Java AWS CloudHSM Cryptography Extension (JCE)

のKeyStoreサービスプロバイダーインターフェイス (SPI) を実装します。KeyStore の使用の詳細については、「Class KeyStore」を参照してください。

> **Note**
>
> 証明書は公開情報であり、暗号化キーのストレージ容量を最大化するため、 AWS CloudHSM は HSMs への証明書の保存をサポートしていません。

AWS CloudHSM クライアント SDK 5 に適したキーストアを選択する

AWS CloudHSM Java Cryptographic Extension (JCE) プロバイダーは、専用 AWS CloudHSM KeyStore を提供しています。クラスは、HSM AWS CloudHSM KeyStoreへのキーオペレーションのオフロード、証明書のローカルストレージ、および証明書ベースのオペレーションをサポートしています。

次のように、特殊目的の CloudHSM KeyStore をロードします。

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

AWS CloudHSM KeyStoreクライアント SDK 5 を初期化する

JCE プロバイダーにログインするのと同じ方法で、 AWS CloudHSM KeyStore にログインします。環境変数またはシステムプロパティファイルを使用できます。CloudHSM KeyStore を使用する前にログインする必要があります。JCE プロバイダーを使用して HSM にログインする例については、Login to an HSM を参照してください。

必要に応じて、パスワードを指定して、キーストアデータを保持するローカル PKCS12 ファイルを暗号化できます。 AWS CloudHSM Keystore を作成するときは、パスワードを設定し、ロード、設定、取得の方法を使用するときに指定します。

新しい CloudHSM KeyStore オブジェクトを次のようにインスタンス化します。

```
ks.load(null, null);
```

store メソッドを使用して、キーストアデータをファイルに書き込みます。その後は、次のように、ソースファイルとパスワードを使用し、load メソッドを使用して既存のキーストアをロードできます。

```
ks.load(inputStream, password);
```

Use AWS CloudHSM KeyStore または AWS CloudHSM クライアント SDK 5

AWS CloudHSM KeyStore は JCE Class KeyStore 仕様に準拠しており、以下の関数を提供します。

- `load`

  指定された入力ストリームからキーストアをロードします。キーストアの保存時にパスワードが設定されている場合、ロードを成功させるには、この同じパスワードを指定する必要があります。新しい空のキーストアを初期化するには、両方のパラメータを null に設定します。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

  指定されたキーストアインスタンス内に含まれるすべてのエントリのエイリアス名の列挙を返します。結果には、PKCS12 ファイルにローカルに保存されたオブジェクトと、HSM 上に存在するオブジェクトが含まれます。

  サンプルコード:

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
      String label = entry.nextElement();
      System.out.println(label);
  }
  ```

- `containsalias`

  キーストアが、指定されたエイリアスを持つ少なくとも 1 つのオブジェクトにアクセスできる場合は true を返します。キーストアは、PKCS12 ファイルにローカルに保存されているオブジェクトと、HSM 上に存在するオブジェクトをチェックします。

- `deleteEntry`

  ローカル PKCS12 ファイルから証明書エントリを削除します。HSM に保存されているキーデータの削除は、 AWS CloudHSM KeyStore ではサポートされていません。Destroyable インターフェースの `destroy` メソッドを使用してキーを削除できます。

  ```
  ((Destroyable) key).destroy();
  ```

- `getCertificate`

使用可能な場合、エイリアスに関連付けられた証明書を返します。エイリアスが存在しないか、証明書ではないオブジェクトを参照している場合、関数は NULL を返します。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias);
```

- `getCertificateAlias`

  指定された証明書とデータが一致する最初のキーストアエントリの名前 (エイリアス) を返します。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert);
```

- `getCertificateChain`

  指定されたエイリアスに関連付けられた証明書チェーンを返します。エイリアスが存在しないか、証明書ではないオブジェクトを参照している場合、関数は NULL を返します。

- `getCreationDate`

  指定されたエイリアスによって識別されるエントリの作成日を返します。作成日が使用できない場合、この関数は証明書が有効になった日付を返します。

- `getKey`

  GetKey が HSM に渡され、指定されたラベルに対応するキーオブジェクトを返します。`getKey` が HSM を直接照会すると、KeyStore によって生成されたかどうかに関係なく、HSM 上の任意のキーに使用できます。

```
Key key = ks.getKey(keyLabel, null);
```

- `isCertificateEntry`

  指定されたエイリアスを持つエントリが証明書エントリを表すかどうかをチェックします。

- `isKeyEntry`

  指定されたエイリアスを持つエントリがキーエントリを表すかどうかをチェックします。このアクションは、PKCS12 ファイルと HSM の両方でエイリアスを検索します。

- `setCertificateEntry`

指定された証明書を指定されたエイリアスに割り当てます。指定されたエイリアスがキーまたは証明書の識別にすでに使用されている場合は、KeyStoreException がスローされます。JCE コードを使用してキーオブジェクトを取得し、KeyStore SetKeyEntry メソッドを使用して証明書をキーに関連付けることができます。

- byte[] キーのある setKeyEntry

  この API は現在、クライアント SDK 5 ではサポートされていません。

- Key オブジェクトのある setKeyEntry

  指定されたキーを指定されたエイリアスに割り当て、HSM 内に保存します。キーが HSM 内にまだ存在しない場合は、抽出可能なセッションキーとして HSM にインポートされます。

  Key オブジェクトが PrivateKey のタイプの場合、対応する証明書チェーンが添付されている必要があります。

  エイリアスが既に存在する場合、SetKeyEntry 呼び出しは KeyStoreException をスローし、キーが上書きされるのを防ぎます。キーを上書きする必要がある場合は、そのために KMU または JCE を使用します。

- engineSize

  キーストア内のエントリの数を返します。

- store

  キーストアを指定された出力ストリームに PKCS12 ファイルとして保存し、指定されたパスワードで保護します。さらに、ロードされたすべてのキー (setKey 呼び出しを使用して設定される) が保持されます。

AWS CloudHSM JCE for Client SDK 5 の詳細設定

AWS CloudHSM JCE プロバイダーには、ほとんどのお客様が使用する一般的な設定の一部ではない、以下の高度な設定が含まれています。

- 複数のクラスターへの接続
- JCE を使用したキー抽出
- JCE の設定を再試行

JCE プロバイダーを使用した複数の AWS CloudHSM クラスターへの接続

この構成では、1 つのクライアントインスタンスが複数の AWS CloudHSM クラスターと通信できます。1 つのインスタンスが 1 つのクラスターとしか通信しない場合と比較して、これは一部のユースケースではコスト削減機能となる可能性があります。`CloudHsmProvider` クラスは、Java セキュリティのプロバイダークラスの AWS CloudHSM実装です。 https://docs.oracle.com/en/java/javase/25/docs/api/java.base/java/security/Provider.htmlこのクラスの各インスタンスは、 AWS CloudHSM クラスター全体への接続を表します。このクラスをインスタンス化して Java セキュリティプロバイダのリストに追加すると、標準 JCE クラスを使用して操作できるようになります。

次の例では、このクラスをインスタンス化して Java セキュリティプロバイダのリストに追加します。

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
    Security.addProvider(new CloudHsmProvider());
}
```

`CloudHsmProvider` は 2 つの方法で設定できます。

1. ファイルによる設定 (デフォルト設定)
2. コードを使用して設定

以下のトピックでは、これらの設定と、複数のクラスターに接続する方法について説明します。

トピック

- ファイルによる AWS CloudHSM CloudHsmProvider クラスの設定 (デフォルト設定)
- コードを使用して AWS CloudHSM CloudHsmProvider クラスを設定する
- 複数の AWS CloudHSM クラスターに接続する

ファイルによる AWS CloudHSM **`CloudHsmProvider`** クラスの設定 (デフォルト設定)

クラスを設定するデフォルトの方法は、 AWS CloudHSM `CloudHsmProvider` ファイルを使用することです。

デフォルトのコンストラクタを使用して `CloudHsmProvider` をインスタンス化すると、デフォルトでは、Linux の `/opt/cloudhsm/etc/cloudhsm-jce.cfg` パスで構成ファイルが検索されます。この設定ファイルは、`configure-jce` を使用して設定できます。

デフォルトコンストラクターを使用して作成されたオブジェクトは、デフォルトの CloudHSM プロバイダー名 CloudHSM を使用します。プロバイダー名は JCE とやり取りして、さまざまなオペレーションにどのプロバイダーを使用するかを判断するのに役立ちます。Cipher オペレーションに CloudHSM プロバイダー名を使用する例は次のとおりです。

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

コードを使用して AWS CloudHSM **CloudHsmProvider** クラスを設定する

クライアント SDK バージョン 5.8.0 以降では、Java コードを使用して クラスを設定 AWS CloudHSM CloudHsmProviderすることもできます。そのための方法は、CloudHsmProviderConfig クラスのオブジェクトを使用することです。CloudHsmProviderConfigBuilder を使用してこのオブジェクトを構築することができます。

CloudHsmProvider には、次の例のように、CloudHsmProviderConfig オブジェクトを取得する別のコンストラクターがあります。

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath)

 .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
                        .build())
        .build();
CloudHsmProvider provider = new CloudHsmProvider(config);
```

この例では、JCE プロバイダーの名前は CloudHsmCluster1 です。これは、アプリケーションが JCE とやり取りするために使用できる名前です。

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

あるいは、アプリケーションは上記で作成したプロバイダーオブジェクトを使用して、そのプロバイダーをオペレーションに使用することを JCE に知らせることもできます。

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

withClusterUniqueIdentifier メソッドで一意の識別子が指定されていない場合は、ランダムに生成されたプロバイダー名が自動的に作成されます。このランダムに生成された識別子を取得するには、アプリケーションは provider.getName() を呼び出して識別子を取得できます。

複数の AWS CloudHSM クラスターに接続する

各 AWS CloudHSM は、クラスターへの接続CloudHsmProviderを表します。同じアプリケーションから別のクラスターと通信したい場合は、別のクラスターの設定を使用して CloudHsmProvider のオブジェクトをもう1つ作成し、次の例に示すように、プロバイダーオブジェクトまたはプロバイダー名を使用してこの他のクラスターとやり取りできます。

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath)

 .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
                        .build())
        .build();
CloudHsmProvider provider1 = new CloudHsmProvider(config);

if (Security.getProvider(provider1.getName()) == null) {
    Security.addProvider(provider1);
}

CloudHsmProviderConfig config2 = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath2)

 .withClusterUniqueIdentifier("CloudHsmCluster2")
        .withServer(CloudHsmServer.builder().withHostIP(hostName2).build())
                        .build())
        .build();
CloudHsmProvider provider2 = new CloudHsmProvider(config2);

if (Security.getProvider(provider2.getName()) == null) {
```

```
    Security.addProvider(provider2);
}
```

上記の両方のプロバイダー (両方のクラスター) を設定したら、プロバイダーオブジェクトまたはプロバイダー名を使用してプロバイダーを操作できます。

cluster1 と通信する方法を示すこの例を拡張すると、AES/GCM/NoPadding オペレーションに次のサンプルを使用できます。

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

同じアプリケーションで、プロバイダー名を使用して 2 番目のクラスターで「AES」キー生成を行う場合は、次のサンプルを使用することもできます。

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

の JCE を使用したキー抽出 AWS CloudHSM

Java Cryptography Extension (JCE) は、さまざまな暗号化実装をプラグインできるアーキテクチャを使用します。 は、暗号化オペレーションを HSM にオフロードする 1 つの JCE プロバイダーを AWS CloudHSM 出荷します。他のほとんどの JCE プロバイダーが AWS CloudHSM に保存されているキーを操作するには、HSM のキーバイトをクリアテキストでマシンのメモリに抽出して使用する必要があります。HSM では通常、キーを ラップされたオブジェクト としてのみ抽出でき、クリアテキストとして抽出することはできません。ただし、プロバイダー間の統合のユースケースをサポートするために、 では、クリアでキーバイトの抽出を有効にするオプトイン設定オプション AWS CloudHSM を許可します。

> **Important**
>
> JCE は、AWS CloudHSM プロバイダーが指定されるか、 AWS CloudHSM キーオブジェクトが使用される AWS CloudHSM たびに、オペレーションを にオフロードします。HSM 内でオペレーションが行われることが予想される場合は、キーを明確に抽出する必要はありません。クリアテキストでのキー抽出が必要なのは、サードパーティのライブラリや JCE プロバイダーの制限により、アプリケーションがキーのラップやラップ解除などの安全なメカニズムを使用できない場合のみです。

AWS CloudHSM JCE プロバイダーは、デフォルトで外部 JCE プロバイダーと連携するパブリックキーの抽出を許可します。以下の方法は常に許可されています。

Class	方法	Format (getEncoded)
EcPublicKey	getEncoded ()	X.509
	getW()	該当なし
RSAPublicKey	getEncoded ()	X.509
	getPublicExponent()	該当なし
CloudHsmEdDSAPublicKey	getEncoded ()	X.509
CloudHsmMldsaPublicKey	getEncoded ()	X.509

AWS CloudHSM JCE プロバイダーは、デフォルトでプライベートキーまたはシークレットキーに対してクリアなキーバイトの抽出を許可しません。ユースケースで必要な場合は、以下の条件で プライベート または シークレット キーのキーバイトを消去して抽出できます。

1. プライベートまたはシークレットキーの EXTRACTABLE 属性は「true」に設定されています。
   - デフォルトでは、プライベートキーとシークレットキーの EXTRACTABLE 属性は「true」に設定されています。EXTRACTABLE キーは HSM からのエクスポートが許可されているキーです。詳細については、「クライアント SDK 5 向けサポートされている Java 属性」を参照してください。
2. プライベートキーとシークレットキーの WRAP_WITH_TRUSTED 属性は「false」に設定されます。
   - getEncoded、getPrivateExponent、getSおよびクリアでエクスポートできないプライベートキーでは使用できません。WRAP_WITH_TRUSTED は、プライベートキーを HSM からクリアにエクスポートすることはできません。詳細については、「信頼できるキーを使ったキーのアンラップの制御」を参照してください。

JCE プロバイダーがプライベートキーシークレットを から抽出することを許可する AWS CloudHSM

次の手順を使用して、 AWS CloudHSM JCE プロバイダーがプライベートキーシークレットを抽出できるようにします。

> **Important**
>
> この設定変更により、HSM クラスターからすべてのクリアの EXTRACTABLE キーバイトを抽出できるようになります。セキュリティを高めるには、キーラッピング方法を使用して HSM から安全にキーを抽出することを検討してください。これにより、HSM からキーバイトが意図せず抽出されるのを防ぐことができます。

1. 以下のコマンドを使用して、プライベート キーと シークレット キーを JCE で抽出できるようにします。

   Linux

   ```
   $ /opt/cloudhsm/bin/configure-jce --enable-clear-key-extraction-in-software
   ```

   Windows

   ```
   PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --enable-clear-key-extraction-in-software
   ```

2. クリアキー抽出を有効にすると、以下の方法でプライベートキーをメモリーに抽出できるようになります。

Class	方法	Format (getEncoded)
Key	getEncoded ()	RAW
ECPrivateKey	getEncoded ()	PKCS#8
	getS()	該当なし
RSAPrivateCrtKey	getEncoded ()	X.509
	getPrivateExponent()	該当なし
	getPrimeP()	該当なし
	getPrimeQ()	該当なし

Class	方法	Format (getEncoded)
	getPrimeExponentP()	該当なし
	getPrimeExponentQ()	該当なし
	getCrtCoefficient()	該当なし
CloudHsmEdDSAPrivateKey	getEncoded ()	PKCS#8
CloudHsmMldsaPrivateKey	getEncoded ()	該当なし

JCE がキーをクリアでエクスポートできないようにして、デフォルトの動作に戻したい場合は、以下のコマンドを実行します。

Linux

```
$ /opt/cloudhsm/bin/configure-jce --disable-clear-key-extraction-in-software
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-clear-key-extraction-in-software
```

の JCE の再試行コマンド AWS CloudHSM

AWS CloudHSM クライアント SDK 5.8.0 以降には、HSM スロットリングされたオペレーションをクライアント側から再試行する自動再試行戦略が組み込まれています。HSM が以前のオペレーションが多すぎてそれ以上リクエストを受け付けられないためにオペレーションをスロットリングすると、Client SDK はスロットリングされたオペレーションを最大 3 回再試行しますが、その間、エクスポネンシャルバックオフします。この自動再試行戦略は、オフとスタンダードの 2 つのモードのいずれかに設定できます。

- オフ: クライアント SDK は、HSM によってスロットリングされたオペレーションに対しては再試行戦略を一切実行しません。
- スタンダード: これはクライアント SDK 5.8.0 以降のデフォルトモードです。このモードでは、クライアント SDK はエクスポネンシャルバックオフすることで、スロットリングされた操作を自動的に再試行します。

詳細については、「HSM スロットリング」を参照してください。

再試行コマンドをオフモードに設定する

Linux

Linux でクライアント SDK 5 向けに再試行コマンドを off に設定するには

- 次のコマンドを使用して再試行設定を off モードに設定できます。

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

Windows 上の クライアント SDK 5 向けに再試行コマンドを off に設定するには

- 次のコマンドを使用して再試行設定を off モードに設定できます。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --default-
retry-mode off
```

以前の SDK バージョンを使用して を操作する AWS CloudHSM

⚠ SDK バージョン 5.8.0 以前のバージョンはサポートが終了しました。2025 年 3 月 31 日以降、SDK バージョン 3.4.4 以前のドキュメントは利用できなくなります。

AWS CloudHSM には、2 つの主要なクライアント SDK バージョンが含まれています。

- クライアント SDK 5: これは最新かつデフォルトの クライアント SDK です。クライアント SDK 5 による利点については、「AWS CloudHSM クライアント SDK 5 の利点」を参照してください。
- クライアント SDK 3: これは古いクライアント SDK です。プラットフォームおよび言語ベースのアプリケーションの互換性および管理ツール用のコンポーネントの完全なセットが含まれています。

クライアント SDK 3 から クライアント SDK 5 に移行する手順については、「AWS CloudHSM クライアント SDK 3 からクライアント SDK 5 への移行」を参照してください。

このトピックでは、クライアント SDK 3 について説明します。使用しているクライアント SDK のバージョンを確認するには、「AWS CloudHSM クライアント SDK のバージョンを確認する」を参照してください。

ダウンロードするには、「ダウンロード」を参照してください。

トピック

- Linux での AWS CloudHSM クライアント SDK 3 のアップグレード
- AWS CloudHSM クライアント SDK 3 でサポートされているプラットフォーム
- AWS CloudHSM クライアント SDK 3 用の PKCS #11 ライブラリ
- AWS CloudHSM クライアント SDK 3 用の OpenSSL Dynamic Engine
- AWS CloudHSM クライアント SDK 3 の JCE プロバイダー
- 暗号化 API: の次世代 (CNG) およびキーストレージプロバイダー (KSP) AWS CloudHSM

Linux での AWS CloudHSM クライアント SDK 3 のアップグレード

⚠ SDK バージョン 5.8.0 以前のバージョンはサポートが終了しました。2025 年 3 月 31 日以降、SDK バージョン 3.4.4 以前のドキュメントは利用できなくなります。

AWS CloudHSM Client SDK 3.1 以降では、アップグレードするには、クライアントデーモンのバージョンとインストールするコンポーネントが一致している必要があります。すべての Linux ベースのシステムでは、1 つのコマンドを使用して、同じバージョンの PKCS #11 ライブラリ、Java 暗号化拡張機能 (JCE) プロバイダー、または OpenSSL Dynamic Engine を使用してクライアントデーモンを一括更新する必要があります。CNG および KSP プロバイダーのバイナリーがすでにクライアントデーモンパッケージに含まれているため、この要件は Windows ベースのシステムには適用されません。

クライアントデーモンバージョンをチェックするには

- Red Hat ベースの Linux システム (Amazon Linux および CentOS を含む) では、次のコマンドを使用します。

```
rpm -qa | grep ^cloudhsm
```

- Debian ベースの Linux システムでは、次のコマンドを使用します。

```
apt list --installed | grep ^cloudhsm
```

- Windows システムでは、次のコマンドを使用します。

```
wmic product get name,version
```

トピック

- 前提条件
- ステップ 1: クライアントデーモンを停止する
- ステップ 2: クライアント SDK をアップグレードする
- ステップ 3: クライアントデーモンを起動する

前提条件

最新バージョンの AWS CloudHSM クライアントデーモンをダウンロードし、コンポーネントを選択します。

> **Note**
>
> すべてのコンポーネントをインストールする必要はありません。インストールしたコンポーネントごとに、クライアントデーモンのバージョンに合わせてコンポーネントをアップグレードする必要があります。

最新の Linux クライアントデーモン

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

最新の PKCS #11 ライブラリ

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

最新の OpenSSL Dynamic Engine

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

最新の JCE プロバイダー

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

ステップ 1: クライアントデーモンを停止する

クライアントデーモンを停止するには、次のコマンドを使用します。

Amazon Linux

```
$ sudo stop cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

ステップ 2: クライアント SDK をアップグレードする

次のコマンドは、クライアントデーモンとコンポーネントのアップグレードに必要な構文を示しています。コマンドを実行する前に、アップグレードしないコンポーネントをすべて削除します。

Amazon Linux

```
$ sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm \
              <./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm> \
              <./cloudhsm-client-dyn-latest.el6.x86_64.rpm> \
              <./cloudhsm-client-jce-latest.el6.x86_64.rpm>
```

Amazon Linux 2

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
              <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
              <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
              <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
              <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
              <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
              <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \
              <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \
              <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

RHEL 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
```

```
                <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

RHEL 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

Ubuntu 16.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_amd64.deb \
                <cloudhsm-client-pkcs11_latest_amd64.deb> \
                <cloudhsm-client-dyn_latest_amd64.deb> \
                <cloudhsm-client-jce_latest_amd64.deb>
```

Ubuntu 18.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb \
                <cloudhsm-client-pkcs11_latest_amd64.deb> \
                <cloudhsm-client-jce_latest_amd64.deb>
```

ステップ 3: クライアントデーモンを起動する

クライアントデーモンを起動するには、以下のコマンドを使用します。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

CentOS 8

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 8

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 20.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

AWS CloudHSM クライアント SDK 3 でサポートされているプラットフォーム

⚠ SDK バージョン 5.8.0 以前のバージョンはサポートが終了しました。2025 年 3 月 31 日以降、SDK バージョン 3.4.4 以前のドキュメントは利用できなくなります。

AWS CloudHSM クライアント SDK 3 にはクライアントデーモンが必要で、CloudHSM 管理ユーティリティ (CMU)、キー管理ユーティリティ (KMU)、 設定ツールなどのコマンドラインツールが用意されています。

基本サポートは、 AWS CloudHSM クライアント SDK のバージョンごとに異なります。通常、SDK 内のコンポーネントのプラットフォームのサポートは基本サポートと一致しますが、必ずしもそう

とは限りません。特定のコンポーネントのプラットフォームのサポートを確認するには、まず目的のプラットフォームが SDK のベースセクションに表示されていることを確認し、コンポーネントセクションで除外項目やその他の関連情報がないか確認します。

プラットフォームのサポートは時間の経過とともに変化します。以前のバージョンの CloudHSM クライアント SDK では、ここに記載されているすべてのオペレーティングシステムがサポートされていない場合があります。リリースノートを使用して、以前のバージョンの CloudHSM クライアント SDK に対するオペレーティングシステムサポートを確認します。詳細については、「AWS CloudHSM クライアント SDK のダウンロード」を参照してください。

AWS CloudHSM は 64 ビットオペレーティングシステムのみをサポートします。

トピック

- AWS CloudHSM クライアント SDK 3 の Linux サポート
- AWS CloudHSM クライアント SDK 3 の Windows サポート
- AWS CloudHSM クライアント SDK 3 の HSM 互換性

AWS CloudHSM クライアント SDK 3 の Linux サポート

AWS CloudHSM クライアント SDK 3 は、次の Linux オペレーティングシステムとプラットフォームをサポートしています。

- Amazon Linux
- Amazon Linux 2
- CentOS 6.10+ [2]
- CentOS 7.3+
- CentOS 8 [1、4]
- Red Hat Enterprise Linux (RHEL) 6.10+ [2]
- Red Hat Enterprise Linux (RHEL) 7.3+
- Red Hat Enterprise Linux (RHEL) 8 [1]
- Ubuntu 16.04 LTS [3]
- Ubuntu 18.04 LTS [1]

[1] OpenSSL Dynamic Engine はサポートされていません。詳細については、「OpenSSL Dynamic Engine」を参照してください。

[2] クライアント SDK 3.3.0 以降はサポートされていません。

[3] SDK 3.4 は Ubuntu 16.04 でサポートされる最後のリリースです。

[4] SDK 3.4 は CentOS 8.3+ でサポートされる最後のリリースです。

AWS CloudHSM クライアント SDK 3 の Windows サポート

AWS CloudHSM クライアント SDK 3 は、次のバージョンの Windows Server をサポートしています。

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

AWS CloudHSM クライアント SDK 3 の HSM 互換性

次の表に、HSMs の AWS CloudHSM クライアント SDK 3 の互換性を示します。

hsm1.medium	hsm2m.medium
クライアントバージョン SDK 3.1.0 以降と互換性があります。	サポート外。

AWS CloudHSM クライアント SDK 3 用の PKCS #11 ライブラリ

PKCS #11 は、 AWS CloudHSMのハードウェア セキュリティ モジュール (HSM) で暗号化オペレーションを実行するための標準です。

ブートストラップの詳細については、「クラスターへの接続」を参照してください。

トピック

- AWS CloudHSM クライアント SDK 3 用の PKCS #11 ライブラリをインストールする
- AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリへの認証
- AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリでサポートされているキータイプ

- AWS CloudHSM クライアント SDK 3 でサポートされているメカニズム
- AWS CloudHSM クライアント SDK 3 でサポートされている API オペレーション
- AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリの主な属性
- AWS CloudHSM クライアント SDK 3 用の PKCS #11 ライブラリのコードサンプル

AWS CloudHSM クライアント SDK 3 用の PKCS #11 ライブラリをインストールする

このトピックでは、 AWS CloudHSM クライアント SDK 3 バージョンシリーズ用の PKCS #11 ライブラリをインストールする手順について説明します。クライアント SDK または PKCS #11 ライブラリの詳細については、[クライアント SDK の使用] と [PKCS #11 ライブラリ] を参照してください。

クライアント SDK 3 の前提条件

PKCS #11 ライブラリには AWS CloudHSM クライアントが必要です。

AWS CloudHSM クライアントをインストールして設定していない場合は、「」のステップに従って実行しますクライアント (Linux) のインストール。クライアントのインストールと設定が完了したら、次のコマンドを使用して起動します。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

CentOS 8

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 8

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 20.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

クライアント SDK 3 用の PKCS #11 ライブラリのインストール

次のコマンドでは、PKCS #11 ライブラリをダウンロードしてインストールします。

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

- PKCS #11 ライブラリをインストールした EC2 インスタンスに、クライアント SDK 3 の他のコンポーネントがインストールされていない場合は、クライアント SDK 3 をブートストラップする必要があります。クライアント SDK 3 のコンポーネントを使用して、各インスタンスで 1 回だけ実行する必要があります。
- PKCS #11 ライブラリのファイルは、次の場所にあります。

  Linuxのバイナリ、設定スクリプト、証明書および口グファイル:

```
/opt/cloudhsm/lib
```

AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリへの認証

PKCS #11 ライブラリを使用すると、アプリケーションは AWS CloudHSMで特定の Crypto User (CU) として実行されます。アプリケーションは、CU が所有して共有するキーのみを表示および管理できます。既存の CU を HSM で使用することも、新しい CU を作成することもできます。CU の管理については、「Managing HSM users with CloudHSM CLI」および「Managing HSM users with CloudHSM Management Utility (CMU)」を参照してください。

PKCS #11 に CU を指定するには、PKCS #11 [C_Login 関数] のピンパラメーターを使用します。の場合 AWS CloudHSM、ピンパラメータの形式は次のとおりです。

```
<CU_user_name>:<password>
```

たとえば、次のコマンドでユーザー名 `CryptoUser` とパスワード `CUPassword123!` を使用して PKCS #11 ライブラリのピンを CU に設定します。

```
CryptoUser:CUPassword123!
```

AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリでサポートされているキータイプ

PKCS #11 ライブラリは、 AWS CloudHSM クライアント SDK 3 で次のキータイプをサポートしています。

キータイプ	説明
RSA	256 ビットの増分で、2,048 ~ 4,096 ビットの RSA キーを生成します。
EC	secp224r1 (P-224)、secp256r1 (P-256)、secp256k1 (ブロックチェーン)、secp384r1 (P-384)、secp521r1 (P-521) のカーブを使用してキーを生成します。
AES	128、192、256 ビットの AES キーを生成します。
3DES (Triple DES)	192 ビットの DES3 キーを生成します。今後の変更については、以下の注記「1」を参照してください。
[GENERIC_SECRET]	1 ~ 64 バイトの汎用シークレットを生成します。

- [1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

AWS CloudHSM クライアント SDK 3 でサポートされているメカニズム

PKCS #11 ライブラリは、 AWS CloudHSM クライアント SDK 3 の次のアルゴリズムをサポートしています。

- [暗号化と復号化] AES-CBC、AES-CTR、AES-ECB、AES-GCM、DES3-CBC、DES3-ECB、RSA-OAEP、RSA-PKCS
- [署名と確認] RSA、HMAC、ECDSA (ハッシュあり、なし)
- [ハッシュ/ダイジェスト] SHA1、SHA224、SHA256、SHA384、SHA512
- [キーラップ] AES キーラップ、[4] AES-GCM、RSA-AES、RSA-OAEP
- [キーの導出] — ECDH、[5] SP800-108 CTR KDF

PKCS #11 ライブラリのメカニズムと関数を示す表

PKCS #11 ライブラリは PKCS #11 仕様のバージョン 2.40 に準拠しています。PKCS#11 を使用して暗号化機能を呼び出すには、指定されたメカニズムで関数を呼び出します。次の表は、 AWS CloudHSMでサポートされている関数とメカニズムの組み合わせをまとめたものです。

サポートされている PKCS#11 メカニズムと関数を示すテーブルの解釈

✓ マークは、 が関数のメカニズム AWS CloudHSM をサポートしていることを示します。PKCS #11 仕様に一覧表示されている利用可能な関数がすべてサポートされているわけではありません。✖ マークは、PKCS #11 標準で許可されている場合でも、 AWS CloudHSM が特定の関数のメカニズムをまだサポートしていないことを示します。空のセルは、PKCS #11 標準で特定の関数のメカニズムがサポートされていないことを示します。

サポートされている PKCS#11 ライブラリのメカニズムと関数

メカニズム	関数						
	キーの生成またはキーペア	署名と検証	SR と VR	ダイジェスト	暗号化と復号	派生キー	ラップとラップ解除
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						
CKM_RSA_X9_31_KEY_PAIR_GEN	✓[2]						
CKM_RSA_X_509		✓			✓		
CKM_RSA_PKCS [注]		✓[1]	✖		✓[1]		✓[1]

メカニズム	関数						
記「8」を参照							
CKM_RSA_PKCS_OAEP					✓1		✓6
CKM_SHA1_RSA_PKCS		✓3.2					
CKM_SHA224_RSA_PKCS		✓3.2					
CKM_SHA256_RSA_PKCS		✓3.2					
CKM_SHA384_RSA_PKCS		✓2,3.2					
CKM_SHA512_RSA_PKCS		✓3.2					
CKM_RSA_PKCS_PSS		✓1					
CKM_SHA1_RSA_PKCS_PSS		✓3.2					
CKM_SHA224_RSA_PKCS_PSS		✓3.2					

メカニズム	関数						
CKM_SHA256_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA384_RSA_PKCS_PSS		✓[2,3.2]					
CKM_SHA512_RSA_PKCS_PSS		✓[3.2]					
CKM_EC_KEY_PAIR_GEN	✓						
CKM_ECDSA		✓[1]					
CKM_ECDSA_SHA1		✓[3.2]					
CKM_ECDSA_SHA224		✓[3.2]					
CKM_ECDSA_SHA256		✓[3.2]					
CKM_ECDSA_SHA384		✓[3.2]					
CKM_ECDSA_SHA512		✓[3.2]					
CKM_ECDH1_DERIVE						✓[5]	

メカニズム	関数						
CKM_SP800_108_COUNTER_KDF						✓	
CKM_GENERIC_SECRET_KEY_GEN	✓						
CKM_AES_KEY_GEN	✓						
CKM_AES_ECB					✓		✖
CKM_AES_CTR					✓		✖
CKM_AES_CBC					✓[3.3]		✖
CKM_AES_CBC_PAD					✓		✖
CKM_DES3_KEY_GEN 注記「8」を参照	✓						
CKM_DES3_CBC 注記「8」を参照					✓[3.3]		✖

メカニズム	関数						
CKM_DES3_CBC_PAD 注記「8」を参照					✓		✖
CKM_DES3_ECB 注記「8」を参照					✓		✖
CKM_AES_GCM					✓[3.3, 4]		✓[7.1]
CKM_CLOUDHSM_AES_GCM					✓[7.1]		✓[7.1]
CKM_SHA_1				✓[3.1]			
CKM_SHA_1_HMAC		✓[3.3]					
CKM_SHA224				✓[3.1]			
CKM_SHA224_HMAC		✓[3.3]					
CKM_SHA256				✓[3.1]			
CKM_SHA256_HMAC		✓[3.3]					

メカニズム	関数						
CKM_SHA384				✓[3.1]			
CKM_SHA384_HMAC		✓[3.3]					
CKM_SHA512				✓[3.1]			
CKM_SHA512_HMAC		✓[3.3]					
CKM_RSA_AES_KEY_WRAP							✓
CKM_AES_KEY_WRAP							✓
CKM_AES_KEY_WRAP_PAD							✓
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD							✓[7.1]

メカニズム	関数						
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD							✓[7.1]

メカニズムの注釈

- [1] シングルパートのオペレーションのみ
- [2] メカニズムは機能的には CKM_RSA_PKCS_KEY_PAIR_GEN のメカニズムと似ていますが、p と q の生成に関してより強力な保証を提供します。
- [3.1] クライアント SDK に基づいてハッシュを異なる方法で AWS CloudHSM アプローチします。クライアント SDK 3 では、ハッシュを行う場所はデータのサイズと、シングルパートオペレーションとマルチパートオペレーションのどちらを使用するかによって異なります。

  クライアント SDK 3 のシングルパートのオペレーション

  表 3.1 に、クライアント SDK 3 の各メカニズムの最大のデータ設定サイズを表します。ハッシュ全体が HSM 内で計算されます。16KB を超えるデータサイズはサポートされません。

  表 3.1 シングルパートオペレーションの最大のデータ設定サイズ

[メカニズム]	[最大データサイズ]
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296
CKM_SHA384	16232
CKM_SHA512	16232

  クライアント SDK 3 のマルチパートオペレーション

16 KB を超えるデータサイズのサポートについては、データサイズによってハッシュが行われる場所が決まります。16 KB 未満のデータバッファは HSM 内でハッシュされます。16 KBからシステムの最大のデータサイズまでのバッファは、ソフトウェアでローカルにハッシュされます。[留意点]: ハッシュ関数は機密情報の暗号化を必要としないため、HSM の外部で安全にコンピューティングすることができます。

- [3.2] クライアント SDK に基づいてハッシュを異なる方法で AWS CloudHSM アプローチします。クライアント SDK 3 では、ハッシュを行う場所はデータのサイズと、シングルパートオペレーションとマルチパートオペレーションのどちらを使用するかによって異なります。

  クライアント SDK 3 のシングルパートのオペレーション

  表 3.2 に、クライアント SDK 3 の各メカニズムの最大のデータ設定サイズを表します。16KB を超えるデータサイズはサポートされません。

  表 3.2 シングルパートオペレーションの最大のデータ設定サイズ

[メカニズム]	[最大データサイズ]
CKM_SHA1_RSA_PKCS	16296
CKM_SHA224_RSA_PKCS	16264
CKM_SHA256_RSA_PKCS	16296
CKM_SHA384_RSA_PKCS	16232
CKM_SHA512_RSA_PKCS	16232
CKM_SHA1_RSA_PKCS_PSS	16296
CKM_SHA224_RSA_PKCS_PSS	16264
CKM_SHA256_RSA_PKCS_PSS	16296
CKM_SHA384_RSA_PKCS_PSS	16232
CKM_SHA512_RSA_PKCS_PSS	16232
CKM_ECDSA_SHA1	16296

[メカニズム]	[最大データサイズ]
CKM_ECDSA_SHA224	16264
CKM_ECDSA_SHA256	16296
CKM_ECDSA_SHA384	16232
CKM_ECDSA_SHA512	16232

クライアント SDK 3 のマルチパートオペレーション

16 KB を超えるデータサイズのサポートについては、データサイズによってハッシュが行われる場所が決まります。16 KB 未満のデータバッファは HSM 内でハッシュされます。16 KBからシステムの最大のデータサイズまでのバッファは、ソフトウェアでローカルにハッシュされます。[留意点]: ハッシュ関数は機密情報の暗号化を必要としないため、HSM の外部で安全にコンピューティングすることができます。

- [3.3] 以下のいずれかのメカニズムを使用してデータを操作する際、データバッファが最大データサイズを超えるとエラーになります。これらのメカニズムでは、すべてのデータ処理が HSM 内で行われる必要があります。次の表は、各メカニズムに設定されている最大データサイズを示します:

  表 3.3 最大のデータ設定サイズ

[メカニズム]	[最大データサイズ]
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224

[メカニズム]	[最大データサイズ]
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280

- [4] AES-GCM の暗号化を実行している際、HSM はアプリケーションからの初期化ベクトル (IV) データを受け入れません。HSM が生成した IV を使用する必要があります。HSM で生成された 12 バイトの IV は、指定した CK_GCM_PARAMS パラメータ構造の pIV 要素が指すメモリ参照に書き込まれます。ユーザーが混乱しないよう、バージョン 1.1.1 以降の PKCS#11 SDK では、AES-GCM 暗号化が初期化されると、pIV はゼロ化されたバッファを指し示すようになっています。
- [5]クライアント SDK 3 のみ。メカニズムは SSL/TLS オフロードのケースをサポートするために実装されており、HSM 内の一部でのみ実行されます。このメカニズムを使用する前に、「の PKCS #11 ライブラリの既知の問題 AWS CloudHSM」の「Issue: ECDH key derivation is executed only partially within the HSM」を参照してください。CKM_ECDH1_DERIVE では、secp521r1 (P-521) カーブはサポートされません。
- [6] 次の CK_MECHANISM_TYPE および CK_RSA_PKCS_MGF_TYPE は、CK_RSA_PKCS_OAEP_PARAMS の CKM_RSA_PKCS_OAEP としてサポートされています:
  - CKG_MGF1_SHA1 を使用する CKM_SHA_1
  - CKG_MGF1_SHA224 を使用する CKM_SHA224
  - CKG_MGF1_SHA256 を使用する CKM_SHA256
  - CKM_MGF1_SHA384 を使用する CKM_SHA384
  - CKM_MGF1_SHA512 を使用する CKM_SHA512
- [7.1] ベンダー定義のメカニズム。CloudHSM ベンダー定義のメカニズムを使用するには、コンパイル時に PKCS #11 アプリケーションに /opt/cloudhsm/include/pkcs11/cloudhsm_pkcs11_vendor_defs.h を含める必要があります。

  CKM_CLOUDHSM_AES_GCM: この独自のメカニズムは、標準 CKM_AES_GCM よりもプログラム的に安全な代替手段です。これは、HSM によって生成された IV を、暗号の初期化中に提供される CK_GCM_PARAMS 構造体に書き戻すのではなく、暗号文の先頭に付加します。このメカニズムは C_Encrypt、C_WrapKey、C_Decrypt、C_UnwrapKey 関数で使用できます。このメカニズムを使用する場合は、CK_GCM_PARAMS 構造体内の pIV 変数を NULL に設定する必要があります。このメカニズムを C_Decrypt および C_UnwrapKey と共に使用する場合、IV は、ラップ解除される暗号文の前に付加されることが想定されます。

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: PKCS #5 パディングを使用する AES キーラップ

CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: ゼロパディングを使用する AES キーラップ

AES キーラップに関する追加情報については、[AES キーラップ] を参照してください。

- [8] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

AWS CloudHSM クライアント SDK 3 でサポートされている API オペレーション

PKCS #11 ライブラリは、 AWS CloudHSM クライアント SDK 3 の次の PKCS #11 API オペレーションをサポートしています。

- `C_CloseAllSessions`
- `C_CloseSession`
- `C_CreateObject`
- `C_Decrypt`
- `C_DecryptFinal`
- `C_DecryptInit`
- `C_DecryptUpdate`
- `C_DeriveKey`
- `C_DestroyObject`
- `C_Digest`
- `C_DigestFinal`
- `C_DigestInit`
- `C_DigestUpdate`
- `C_Encrypt`
- `C_EncryptFinal`
- `C_EncryptInit`
- `C_EncryptUpdate`
- `C_Finalize`
- `C_FindObjects`
- `C_FindObjectsFinal`

- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal
- C_SignInit
- C_SignRecover(クライアント SDK 3 のサポートのみ)
- C_SignRecoverInit(クライアント SDK 3 のサポートのみ)
- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyRecover(クライアント SDK 3 のサポートのみ)
- C_VerifyRecoverInit(クライアント SDK 3 のサポートのみ)

- C_VerifyUpdate
- C_WrapKey

AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリの主な属性

キーオブジェクトには、パブリックキー、プライベートキー、またはシークレットキーを指定できます。キーオブジェクトで許可されているアクションは属性で指定されます。属性は、キーオブジェクトの作成時に定義されます。PKCS #11 ライブラリを に使用すると AWS CloudHSM、PKCS #11 標準で指定されたデフォルト値が割り当てられます。

AWS CloudHSM は、PKCS #11 仕様に記載されているすべての属性をサポートしているわけではありません。サポートするすべての属性の仕様に準拠しています。これらの属性は、それぞれのテーブルにリストされています。

オブジェクトを作成、変更、またはコピーする C_CreateObject、C_GenerateKey、C_GenerateKeyPair、C_UnwrapKey、C_DeriveKey などの暗号化関数は、属性テンプレートをパラメータの 1 つとして使用します。オブジェクトの作成中に属性テンプレートを渡す方法の詳細については、「Generate keys through PKCS #11 library」のサンプルを参照してください。

以下のトピックでは、クライアント SDK 3 の AWS CloudHSM キー属性について詳しく説明します。

トピック

- AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリ属性テーブル
- AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリ属性の変更
- AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリエラーコードの解釈

AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリ属性テーブル

AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリテーブルには、キータイプによって異なる属性のリストが含まれています。これは、特定の暗号化関数を で使用するときに、特定のキータイプで特定の属性がサポートされるかどうかを示します AWS CloudHSM。

凡例:

- ✓ CloudHSM が特定のキータイプの属性をサポートしていることを示します。
- ✖ CloudHSM が特定のキータイプの属性をサポートしていないこと示します。

- R は、属性値が特定のキータイプに対して読み取り専用に設定されていることを示します。
- S は、属性が機密であるため、`GetAttributeValue` で読み取れないことを示します。
- [Default Value] 列のセルが空の場合は、属性に割り当てられている特定のデフォルト値がないことを示します。

GenerateKeyPair

属性	キータイプ				デフォルト値
	EC プライベート	EC パブリック	RSA プライベート	RSA パブリック	
CKA_CLASS	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	正
CKA_TOKEN	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✓	✖	✓	誤
CKA_DECRYPT	✓	✖	✓	✖	誤

属性	キータイプ				デフォルト値
CKA_DERIVE	✓	✓	✓	✓	誤
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	誤
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	
CKA_VERIFY	✖	✓	✖	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	
CKA_WRAP	✖	✓	✖	✓	誤
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	
CKA_TRUSTED	✖	✓	✖	✓	誤
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	誤
CKA_UNWRAP	✓	✖	✓	✖	誤

属性	キータイプ				デフォルト値
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	正
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	正
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	
CKA_PRIME_1	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	

属性	キータイプ				デフォルト値
CKA_EXPONENT_2	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

属性	キータイプ			デフォルト値
	AES	DES3	汎用シークレット	
CKA_CLASS	✓	✓	✓	

属性	キータイプ			デフォルト値
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	正
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	誤
CKA_DECRYPT	✓	✓	✖	誤
CKA_DERIVE	✓	✓	✓	誤
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	True
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	True

属性	キータイプ			デフォルト値
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	誤
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	誤
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	誤
CKA_UNWRAP	✓	✓	✖	誤
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	✖	✖	✖	
CKA_EXTRACTABLE	✓	✓	✓	正
CKA_NEVER_EXTRACTABLE	R	R	R	

属性	キータイプ			デフォルト値
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	

属性	キータイプ				デフォルト値
CKA_VALUE_LEN		✓[2]	✖	✓[2]	
CKA_CHECK_VALUE		R	R	R	

CreateObject

属性	キータイプ							デフォルト値
	EC プライベート	EC パブリック	RSA プライベート	RSA パブリック	AES	DES3	汎用シークレット	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	誤
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	誤
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True

属性	キータイプ							デフォルト値
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖	誤
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖	誤
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	誤
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	誤
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	✖	✖	✖	誤
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	誤
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	誤

属性	キータイプ							デフォルト値
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	誤
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	誤
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	正
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	正
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✓	✖	✖	✖	✖	

属性	キータイプ							デフォルト値
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_EC_PARAMS	✓[2]	✓[2]	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✓[2]	✖	✖	✖	✖	✖	
CKA_VALUE	✓[2]	✖	✖	✖	✓[2]	✓[2]	✓[2]	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

属性	キータイプ					デフォルト値
	EC プライベート	RSA プライベート	AES	DES3	汎用シークレット	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	誤
CKA_TOKEN	✓	✓	✓	✓	✓	誤
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✓	✓	✖	誤
CKA_DECRYPT	✖	✓	✓	✓	✖	誤
CKA_DERIVE	✓	✓	✓	✓	✓	誤
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True

属性	キータイプ					デフォルト値
CKA_DESTROYABLE	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	✓	✓	誤
CKA_SIGN_RECOVER	✖	✓[3]	✖	✖	✖	誤
CKA_VERIFY	✖	✖	✓	✓	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	誤
CKA_UNWRAP	✖	✓	✓	✓	✖	誤
CKA_SENSITIVE	✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	正
CKA_NEVER_EXTRACTABLE	R	R	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	
CKA_MODULUS	✖	✖	✖	✖	✖	

属性	キータイプ					デフォルト値
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	✖	

属性	キータイプ					デフォルト値
CKA_VALUE_LEN		✖	✖	✖	✖	✖	
CKA_CHECK_VALUE		R	R	R	R	R	

DeriveKey

属性	キータイプ				デフォルト値
	AES	DES3	汎用シークレット	
CKA_CLASS	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	正
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	誤
CKA_DECRYPT	✓	✓	✖	誤

属性	キータイプ			デフォルト値
CKA_DERIVE	✓	✓	✓	誤
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	True
CKA_SIGN	✓	✓	✓	誤
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	誤
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	誤
CKA_UNWRAP	✓	✓	✖	誤
CKA_SENSITIVE	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	正
CKA_NEVER_EXTRACTABLE	R	R	R	

属性	キータイプ			デフォルト値
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	

属性	キータイプ			デフォルト値
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

属性	キータイプ						
	EC プライベート	EC パブリック	RSA プライベート	RSA パブリック	AES	DES3	汎用シークレット
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓

属性	キータイプ						
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓

属性	キータイプ						
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖

属性	キータイプ						
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓[2]	✓[2]	✓[2]
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

属性注釈

- [1] この属性はファームウェアによって部分的にサポートされており、デフォルト値にのみ明示的に設定する必要があります。

- [2] 必須属性
- [3]クライアント SDK 3 のみ。CKA_SIGN_RECOVER の属性は CKA_SIGN の属性から派生します。設定される場合は、CKA_SIGN に設定されている値と同じ値にのみ設定できます。設定されない場合、CKA_SIGN のデフォルト値が導出されます。CloudHSM では RSA ベースの回復可能な署名メカニズムのみがサポートされるため、この属性は現在 RSA パブリックキーのみに適用されます。
- [4]クライアント SDK 3 のみ。CKA_VERIFY_RECOVER の属性は CKA_VERIFY の属性から派生します。設定される場合は、CKA_VERIFY に設定されている値と同じ値にのみ設定できます。設定されない場合、CKA_VERIFY のデフォルト値が導出されます。CloudHSM では RSA ベースの回復可能な署名メカニズムのみがサポートされるため、この属性は現在 RSA パブリックキーのみに適用されます。

AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリ属性の変更

オブジェクトの属性には、オブジェクトが作成された後に変更できるものもありますが、変更できないものもあります。属性を修正するには、cloudhsm_mgmt_util の[setAttribute] コマンドを使用します。また、cloudhsm_mgmt_util の listAttribute コマンドを使用して属性一覧とそれを表す定数を取得することも可能です。

次のリストで、オブジェクトの作成後に変更が許可荒れている許可されている属性が表示されます。

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 変更が許可されるには、セッションキーをトークンキーに変更する場合のみです。key_mgmt_util の setAttribute コマンドを使用して属性値を変更します。

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP

- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> この属性ではキー取得がサポートされています。すべてのパブリックキーで False を指定する必要があります。True に設定することはできません。シークレットキーまたは EC プライベートキーに対しては、True または False に設定できます。

- CKA_TRUSTED

> **Note**
>
> この属性は Crypto Officer (CO) のみによって True または False に設定できます。

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> この属性をエクスポート可能なデータキーに適用して、このキーを CKA_TRUSTED としてマークされたキーでのみラップできるように指定します。1度 CKA_WRAP_WITH_TRUSTED を true に設定すると属性は読み取り専用になり、属性を変更または削除することはできません。

AWS CloudHSM クライアント SDK 3 の PKCS #11 ライブラリエラーコードの解釈

特定のキーでサポートされていない PKCS #11 ライブラリ属性をテンプレートで指定すると、エラーが発生します。次の表には、仕様に違反した場合に生成されるエラーコードが含まれています。

エラーコード	説明
CKR_TEMPLATE_INCONSISTENT	PKCS#11 仕様に準拠しているが、CloudHSM でサポートされていない属性を属性テンプレートで指定した場合に、このエラーが発生します。

エラーコード	説明
CKR_ATTRIBUTE_TYPE_INVALID	PKCS#11 仕様に準拠しているが、CloudHSM でサポートされていない属性の値を取得すると、このエラーが発生します。
CKR_ATTRIBUTE_INCOMPLETE	このエラーは、属性テンプレートで必須属性を指定しなかった場合に発生します。
CKR_ATTRIBUTE_READ_ONLY	このエラーは、属性テンプレートで読み取り専用属性を指定した場合に発生します。

AWS CloudHSM クライアント SDK 3 用の PKCS #11 ライブラリのコードサンプル

GitHub のコードサンプルは、 AWS CloudHSM用の PKCS #11 ライブラリを使用して基本的なタスクを実行する方法を示しています。

サンプルコードの前提条件

サンプルを実行する前に、以下のステップを実行して環境をセットアップします。

- クライアント SDK 3 用の PKCS #11 ライブラリ のインストールと設定をします。
- 暗号化ユーザー (CU) の設定をします。アプリケーションは、この HSM アカウントを使用して HSM でコードサンプルを実行します。

コードサンプル

PKCS#11 用 AWS CloudHSM ソフトウェアライブラリのコードサンプルは、GitHub で入手できます。このリポジトリには、暗号化、復号化、署名、検証など、PKCS #11 を使用して一般的な操作を行う方法の例が含まれています。

- キーの生成 (AES、RSA、EC)
- キー属性のリスト化
- AES GCM を使用したデータの暗号化および復号
- AES_CTR を使用したデータの暗号化および復号
- 3DES を使用したデータの暗号化および復号
- RSAを使用したデータの署名と検証

- HMAC KDFを使用したキーの取得
- PKCS #5 パディングありの AES を使用したキーのラップとラップ解除
- パディングなしの AES を使用したキーのラップとラップ解除
- ゼロパディングありの AES を使用したキーのラップとラップ解除
- AES-GCM を使用したキーのラップとラップ解除
- RSA を使用したキーのラップとラップ解除

AWS CloudHSM クライアント SDK 3 用の OpenSSL Dynamic Engine

AWS CloudHSM OpenSSL Dynamic Engine を使用すると、OpenSSL API を使用して暗号化オペレーションを CloudHSM OpenSSL クラスターにオフロードできます。

AWS CloudHSM クライアント SDK 3 では、クラスターに接続するためにクライアントデーモンが必要です。以下をサポートします。

- 2048、3072、および 4096 ビットキーの RSA キーの生成。
- RSA の署名/検証。
- RSA の暗号化/復号。
- 暗号化された安全で FIPS 検証済みの乱数生成。

以下のセクションを使用して、OpenSSL 用の AWS CloudHSM 動的エンジンをインストールして設定します。

トピック

- AWS CloudHSM クライアント SDK 3 を使用した OpenSSL Dynamic Engine の前提条件
- AWS CloudHSM クライアント SDK 3 用の OpenSSL Dynamic Engine をインストールする
- AWS CloudHSM クライアント SDK 3 に OpenSSL Dynamic Engine を使用する

AWS CloudHSM クライアント SDK 3 を使用した OpenSSL Dynamic Engine の前提条件

サポートされるプラットフォームの詳細については、「AWS CloudHSM クライアント SDK 3 でサポートされているプラットフォーム」を参照してください。

クライアント SDK 3 で OpenSSL の AWS CloudHSM 動的エンジンを使用するには、 AWS CloudHSM クライアントが必要です。

このクライアントは、クラスター内の HSM とのエンドツーエンドの暗号化された通信を確立するデーモンであり、OpenSSL エンジンはこのクライアントとローカルに通信します。 AWS CloudHSM クライアントをインストールして設定するには、「」を参照してくださいクライアント (Linux) のインストール。次のコマンドを使用して起動します。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 6

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 6

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

AWS CloudHSM クライアント SDK 3 用の OpenSSL Dynamic Engine をインストールする

次の手順では、クライアント SDK 3 で OpenSSL の AWS CloudHSM 動的エンジンをインストールして設定する方法について説明します。アップグレードの詳細については、「クライアント SDK 3 をアップグレードする」を参照してください。

OpenSSL エンジンをインストールして設定するには

1. 以下のコマンドを使用して、OpenSSL エンジンをダウンロードしてインストールします。

   Amazon Linux

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

   Amazon Linux 2

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
   ```

   CentOS 6

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

   CentOS 7

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
   ```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 6

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-dyn_latest_amd64.deb
```

OpenSSL エンジンは `/opt/cloudhsm/lib/libcloudhsm_openssl.so` にインストールされています。

2. 次のコマンドを使用して、Crypto User (CU) の認証情報が含まれている `n3fips_password` という名前の環境変数を設定します。

```
$ export n3fips_password=<HSM user name>:<password>
```

AWS CloudHSM クライアント SDK 3 に OpenSSL Dynamic Engine を使用する

OpenSSL 統合アプリケーションから OpenSSL の AWS CloudHSM 動的エンジンを使用するには、アプリケーションが という名前の OpenSSL 動的エンジンを使用しているこ

とを確認しますcloudhsm。動的エンジンの共有ライブラリは /opt/cloudhsm/lib/libcloudhsm_openssl.so にあります。

OpenSSL コマンドラインから OpenSSL の AWS CloudHSM 動的エンジンを使用するには、 -engineオプションを使用して、 という名前の OpenSSL 動的エンジンを指定しますcloudhsm。例：

```
$ openssl s_server -cert <server.crt> -key <server.key> -engine cloudhsm
```

AWS CloudHSM クライアント SDK 3 の JCE プロバイダー

AWS CloudHSM JCE プロバイダーは、Java 暗号化拡張 (JCE) プロバイダーフレームワークから構築されたプロバイダー実装です。JCE では、Java 開発キット (JDK) を使用して暗号化操作を実行できます。このガイドでは、 AWS CloudHSM JCE プロバイダーは JCE プロバイダーと呼ばれることがあります。JCE プロバイダーと JDK を使用して、HSM に暗号化操作をオフロードします。

トピック

- AWS CloudHSM クライアント SDK 3 の JCE プロバイダーをインストールする
- AWS CloudHSM クライアント SDK 3 の JCE プロバイダーにおけるキー管理の基本
- Client SDK 3 for Client AWS CloudHSM SDK 3 でサポートされているメカニズム
- AWS CloudHSM クライアント SDK 3 でサポートされている Java キー属性
- Java for Client SDK 3 用の AWS CloudHSM ソフトウェアライブラリのコードサンプル
- クライアント SDK 3 のAWS CloudHSM KeyStore Java クラス

AWS CloudHSM クライアント SDK 3 の JCE プロバイダーをインストールする

JCE プロバイダーを使用する前に、 AWS CloudHSM クライアントが必要です。

このクライアントは、クラスターの HSM とエンドツーエンドの暗号化された通信を確立するデーモンです。JCE プロバイダは、クライアントとローカルに通信します。 AWS CloudHSM クライアントパッケージをインストールして設定していない場合は、「」のステップに従って実行しますクライアント (Linux) のインストール。クライアントのインストールと設定が完了したら、次のコマンドを使用して起動します。

JCE プロバイダーは、Linux および互換性のあるオペレーティングシステム上でのみサポートされています。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

CentOS 8

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 8

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 20.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

以下のセクションを使用して、 プロバイダーに認証情報をインストール、検証、提供します。

トピック

- ステップ 1: JCE プロバイダーをインストールする
- ステップ 2: インストールを確認する
- ステップ 3: JCE プロバイダーに認証情報を提供する

ステップ 1: JCE プロバイダーをインストールする

以下のコマンドを使用して、JCE プロバイダーをダウンロードし,インストールします。このプロバイダーは、Linux および互換性のあるオペレーティングシステムでのみサポートされています。

> **Note**
>
> アップグレードについては、「クライアント SDK 3 をアップグレードする」を参照してください。

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_u18.04_amd64.deb
```

前述のコマンドを実行すると、次の JCE プロバイダーファイルが表示されます。

- /opt/cloudhsm/java/cloudhsm-*<version>*.jar

- /opt/cloudhsm/java/cloudhsm-test-*<version>*.jar
- /opt/cloudhsm/java/hamcrest-all-1.3.jar
- /opt/cloudhsm/java/junit.jar
- /opt/cloudhsm/java/log4j-api-2.17.1.jar
- /opt/cloudhsm/java/log4j-core-2.17.1.jar
- /opt/cloudhsm/lib/libcaviumjca.so

ステップ 2: インストールを確認する

インストールを検証するには、HSM で基本的なオペレーションを実行します。

JCE プロバイダーのインストールを検証するには

1. (オプション) 使用環境に Java がインストール済みでない場合は、次のコマンドを使用してインストールします。

   Linux (and compatible libraries)

   ```
   $ sudo yum install java-1.8.0-openjdk
   ```

   Ubuntu

   ```
   $ sudo apt-get install openjdk-8-jre
   ```

2. 次のコマンドを使用して、必要な環境変数を設定します。*<HSM user name>* と *<password>* では、Crypto User (CU) の認証情報に置き換えます。

   ```
   $ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
   ```

   ```
   $ export HSM_PARTITION=PARTITION_1
   ```

   ```
   $ export HSM_USER=<HSM user name>
   ```

   ```
   $ export HSM_PASSWORD=<password>
   ```

3. 基本的な機能のテストを実行するには、次のコマンドを使用します。成功すると、コマンドの出力は次のようになります。

```
$ java8 -classpath "/opt/cloudhsm/java/*" org.junit.runner.JUnitCore
 TestBasicFunctionality

JUnit version 4.11
.2018-08-20 17:53:48,514 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:33) - Adding provider.
2018-08-20 17:53:48,612 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:42) - Logging in.
2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:104) -
 Looking for credentials in HsmCredentials.properties
2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:122) -
 Looking for credentials in System.properties
2018-08-20 17:53:48,613 INFO [main] cfm2.LoginManager (LoginManager.java:130) -
 Looking for credentials in System.env
 SDK Version: 2.03
2018-08-20 17:53:48,655 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:54) - Generating AES Key with key size 256.
2018-08-20 17:53:48,698 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:63) - Encrypting with AES Key.
2018-08-20 17:53:48,705 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:84) - Deleting AES Key.
2018-08-20 17:53:48,707 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:92) - Logging out.

Time: 0.205

OK (1 test)
```

ステップ 3: JCE プロバイダーに認証情報を提供する

HSM では、アプリケーションがそれらを使用する前に、Java アプリケーションを認証する必要があります。アプリケーションごとに 1 つのセッションを使用できます。HSM は、明示的なログインと暗黙的なログイン方法のいずれかを使用して、セッションを認証します。

Explicit login - この方法では、CloudHSM 認証情報をアプリケーションに直接渡すことができます。また、`LoginManager.login()` メソッドを使用します。ここで、CU ユーザー名、パスワード、HSM パーティション ID を渡します。明示的なログイン方法の使用の詳細については、「HSM へのログイン」のサンプルコードを参照してください。

Implicit login - この方法では、CloudHSM 認証情報を、新しいプロパティファイルまたはシステムプロパティで設定するか、環境変数として設定することができます。

- New property file - `HsmCredentials.properties` という名前の新しいファイルを作成し、そのファイルをアプリケーションの CLASSPATH に追加します。ファイルには次の内容が含まれます。

```
HSM_PARTITION = PARTITION_1
HSM_USER = <HSM user name>
HSM_PASSWORD = <password>
```

- System properties - アプリケーションの実行時に、システムプロパティを通して認証情報を設定します。次の例は、これを行うための 2 つの異なる方法を示しています。

```
$ java -DHSM_PARTITION=PARTITION_1 -DHSM_USER=<HSM user name> -
DHSM_PASSWORD=<password>
```

```
System.setProperty("HSM_PARTITION","PARTITION_1");
System.setProperty("HSM_USER","<HSM user name>");
System.setProperty("HSM_PASSWORD","<password>");
```

- Environment variables - 認証情報を環境変数として設定します。

```
$ export HSM_PARTITION=PARTITION_1
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

アプリケーションで設定されない場合、または HSM でセッションを認証する前にユーザーが操作を行った場合は、認証情報を使用できない場合があります。このような場合は、Java 用の CloudHSM ソフトウェアライブラリによって、次の順序で認証情報が検索されます。

1. `HsmCredentials.properties`
2. システムプロパティ
3. 環境変数

エラー処理

暗黙的なログインよりも明示的なログインの方が、簡単にエラーを処理することができます。`LoginManager` クラスを使用すると、アプリケーションが障害に対応する方法をより細かく制御できます。暗黙的なログイン方法では、認証情報が無効な場合や、HSM でのセッションの認証に問題が発生したタイミングをエラー処理で把握するのが難しくなります。

AWS CloudHSM クライアント SDK 3 の JCE プロバイダーにおけるキー管理の基本

JCE プロバイダー中のキー管理の基本には、キーのインポート、キーのエクスポート、ハンドルによるキーのロード、またはキーの削除などがあります。キーの管理の詳細については、「キーの管理」のサンプルコードを参照してください。

また、JCE プロバイダーのサンプルコードについては、コードサンプル で参照できます。

Client SDK 3 for Client AWS CloudHSM SDK 3 でサポートされているメカニズム

このトピックでは、 AWS CloudHSM クライアント SDK 3 で JCE プロバイダーがサポートするメカニズムについて説明します。でサポートされている Java 暗号化アーキテクチャ (JCA) インターフェイスとエンジンクラスの詳細については AWS CloudHSM、以下のトピックを参照してください。

トピック

- サポートされるキー
- サポートされる暗号
- サポートされているダイジェスト
- サポートされている Hash-based Message Authentication Code (HMAC) アルゴリズム
- サポートされている署名/検証メカニズム
- メカニズムの注釈

サポートされるキー

Java 用の AWS CloudHSM ソフトウェアライブラリでは、次のキータイプを生成できます。

- AES – 128、192、256 ビットの AES キー。
- DESede — 92 ビット 3DES キー。今後の変更については、以下の注記「1」を参照してください。
- NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384)、および secp256k1 (ブロックチェーン) を対象とした ECC キーペア。
- RSA – 2,048 ~ 4,096 ビットの RSA キー (256 ビットの増分)。

標準のパラメータに加えて、生成されるキーごとに以下のパラメータがサポートされています。

- Label: キーの検索に使用できるキーラベル。
- isExtractable: キーを HSM からエクスポートできるかどうかを示します。

- isPersistent: 現在のセッションの終了後、キーが HSM に残るかどうかを示します。

> **Note**
>
> Java ライブラリバージョン 3.1 では、パラメータをより詳細に指定することができます。詳細については、「サポートされている Java 属性」を参照してください。

サポートされる暗号

Java 用 AWS CloudHSM ソフトウェアライブラリは、次のアルゴリズム、モード、パディングの組み合わせをサポートしています。

アルゴリズム	モード	[Padding] (パディング)	注意事項
AES	CBC	`AES/CBC/NoPadding` `AES/CBC/PKCS5Padding`	`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。
AES	ECB	`AES/ECB/NoPadding` `AES/ECB/PKCS5Padding`	`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。AES 変換を使用します。
AES	CTR	`AES/CTR/NoPadding`	`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。
AES	GCM	`AES/GCM/NoPadding`	`Cipher.ENCRYPT_MODE` および `Cipher.DE`

アルゴリズム	モード	[Padding] (パディング)	注意事項
			CRYPT_MODE 、Cipher.WRAP_MODE および Cipher.UNWRAP_MODE を実装します。 AES-GCM 暗号化の実行時に、HSM はリクエスト内の初期化ベクトル (IV) を無視し、独自に IV を生成して使用します。オペレーションが完了したら、Cipher.getIV() を呼び出して IV を取得する必要があります。
AESWrap	ECB	AESWrap/ECB/ZeroPadding AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding	Cipher.WRAP_MODE および Cipher.UNWRAP_MODE を実装します。AES 変換 を使用します。

アルゴリズム	モード	[Padding] (パディング)	注意事項

DESede (Triple DES)	CBC	`DESede/CBC/NoPadding`
		
		`DESede/CBC/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。
			
			キー生成ルーチンは 168 ビットまたは 192 ビットのサイズを受け入れます。ただし、内部的に、すべての DESede キーは 192 ビットです。
			
			今後の変更については、以下の注記「1」を参照してください。

アルゴリズム	モード	[Padding] (パディング)	注意事項

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。
			
			キー生成ルーチンは 168 ビットまたは 192 ビットのサイズを受け入れます。ただし、内部的に、すべての DESede キーは 192 ビットです。
			
			今後の変更については、以下の注記「1」を参照してください。

RSA	ECB	`RSA/ECB/NoPadding`
		
		`RSA/ECB/PKCS1Padding`
			`Cipher.ENCRYPT_MODE` および `Cipher.DECRYPT_MODE` を実装します。
			
			今後の変更については、以下の注記「1」を参照してください。

アルゴリズム	モード	[Padding] (パディング)	注意事項
RSA	ECB	RSA/ECB/OAEPPadding RSA/ECB/OAEPWithSHA-1ANDMGF1Padding RSA/ECB/OAEPWithSHA-224ANDMGF1Padding RSA/ECB/OAEPWithSHA-256ANDMGF1Padding RSA/ECB/OAEPWithSHA-384ANDMGF1Padding RSA/ECB/OAEPWithSHA-512ANDMGF1Padding	Cipher.ENCRYPT_MODE 、Cipher.DECRYPT_MODE 、Cipher.WRAP_MODE 、および Cipher.UNWRAP_MODE を実装します。 OAEPPadding は、SHA-1 パディングタイプの OAEP です。
RSAAESWrap	ECB	OAEPPADDING	Cipher.WRAP_Mode および Cipher.UNWRAP_MODE を実装します。

サポートされているダイジェスト

Java 用 AWS CloudHSM ソフトウェアライブラリは、次のメッセージダイジェストをサポートしています。

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

> **Note**
>
> 16 KB 未満のデータは HSM でハッシュされ、それ以上のデータはソフトウェアでローカルにハッシュされます。

サポートされている Hash-based Message Authentication Code (HMAC) アルゴリズム

Java 用の AWS CloudHSM ソフトウェアライブラリは、次の HMAC アルゴリズムをサポートしています。

- HmacSHA1
- HmacSHA224
- HmacSHA256
- HmacSHA384
- HmacSHA512

サポートされている署名/検証メカニズム

Java 用 AWS CloudHSM ソフトウェアライブラリは、次のタイプの署名と検証をサポートしています。

RSA 署名タイプ

- NONEwithRSA
- SHA1withRSA

- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

ECDSA 署名タイプ

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

メカニズムの注釈

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

AWS CloudHSM クライアント SDK 3 でサポートされている Java キー属性

このトピックでは、Java ライブラリバージョン 3.1 の独自の拡張機能を使用して、 AWS CloudHSM クライアント SDK 3 のキー属性を設定する方法について説明します。この拡張機能を使用して、これらのオペレーション中にサポートされるキー属性とその値を設定します。

- キー生成
- キーのインポート
- キーのラップ解除

> **Note**
>
> カスタムキー属性を設定するための拡張機能は、オプション機能です。Java ライブラリバージョン 3.0 で機能するコードがすでにある場合、そのコードを変更する必要はありません。作成したキーには、以前と同じ属性が引き続き含まれます。

トピック

- 属性について
- サポートされている 属性
- キーの属性設定
- まとめ

属性について

キー属性を使用して、パブリックキー、プライベートキー、シークレットキーなど、キーオブジェクトで許可されるアクションを指定します。キー属性と値は、キーオブジェクトの作成オペレーション中に定義します。

ただし、Java Cryptography Extension (JCE) では、キー属性に値を設定する方法が指定されていないため、ほとんどのアクションがデフォルトで許可されていました。これに対して、PKCS # 11 標準では、より制限の厳しいデフォルトのある包括的な属性のセットが定義されています。Java ライブラリバージョン 3.1 以降、CloudHSM は、一般的に使用される属性に対してより制限の厳しい値を設定できる独自の拡張機能を提供します。

サポートされている 属性

次の表に示す属性の値を設定できます。ベストプラクティスとして、制限する属性の値のみを設定してください。値を指定しない場合、CloudHSM は次の表で指定されたデフォルト値を使用します。デフォルト値の列のセルが空の場合は、属性に割り当てられている特定のデフォルト値がないことを示します。

属性	デフォルト値		注意事項
	対称キー	キーペアのパブリックキー	キーペアのプライベートキー	

属性	デフォルト値			注意事項
CKA_TOKEN	FALSE	FALSE	FALSE	クラスター内のすべての HSM にレプリケートされ、バックアップに含まれる永続的なキー。CKA_TOKEN = FALSE は、セッションキーを意味します。セッションキーは 1 つの HSM にのみロードされ、HSM への接続が切断されると自動的に消去されます。
CKA_LABEL				ユーザー定義の文字列。これにより、HSM のキーを簡単に識別できます。
CKA_EXTRACTABLE	TRUE		TRUE	True は、このキーを HSM からエクスポートできることを示します。
CKA_ENCRYPT	TRUE	TRUE		True は、キーを使用して任意のバッファを暗号化できることを示します。

属性	デフォルト値			注意事項
CKA_DECRYPT	TRUE		TRUE	True は、キーを使用して任意のバッファを復号できることを示します。通常は、CKA_WRAP が true に設定されているキーに対して、これを FALSE に設定します。
CKA_WRAP	TRUE	TRUE		True は、キーを使用して別のキーをラップできることを示します。通常、プライベートキーの場合、これを FALSE に設定します。
CKA_UNWRAP	TRUE		TRUE	True は、キーを使用して別のキーをラップ解除 (インポート) できることを示します。

属性	デフォルト値			注意事項
CKA_SIGN	TRUE		TRUE	True は、キーを使用してメッセージダイジェストに署名できることを示します。パブリックキーおよびアーカイブしたプライベートキーの場合、通常 FALSE に設定されます。
CKA_VERIFY	TRUE	TRUE		True は、キーを使用して署名を検証できることを示します。これは通常、プライベートキーの場合、FALSE に設定されます。

属性	デフォルト値			注意事項
CKA_PRIVATE	TRUE	TRUE	TRUE	True は、ユーザーが認証されるまで、ユーザーがキーにアクセスできないことを示します。わかりやすくするために、この属性が FALSE に設定されている場合でも、ユーザーが認証されるまで CloudHSM のどのキーにもアクセスできません。

> **Note**
>
> PKCS #11 ライブラリでは、より広範な属性がサポートされます。詳細については、「サポートされている PKCS #11 属性」を参照してください。

キーの属性設定

CloudHsmKeyAttributesMap は Java Map のようなオブジェクトで、キーオブジェクトの属性値を設定するために使用できます。CloudHsmKeyAttributesMap 関数のメソッドは、Java マップ操作のメソッドと同様です。

属性にカスタム値を設定するには、次の 2 つのオプションがあります。

- 次の表に示す方法を使用します。
- このドキュメントの後半で説明するビルダーパターンの使用

属性マップオブジェクトは、属性を設定するための次のメソッドをサポートしています。

運用	戻り値	**CloudHSMKeyAttributesMap** 方法
既存のキーのキー属性の値を取得する	オブジェクト (値を含む) または null	get(keyAttribute)
1 つのキー属性の値を入力します。	キー属性のマッピングがなかった場合、キー属性に関連付けられた以前の値、または null	put(keyAttribute, value)
複数のキー属性の値を設定する	該当なし	putAll(keyAttributesMap)
属性マップからキーと値のペアを削除する	キー属性のマッピングがなかった場合、キー属性に関連付けられた以前の値、または null	remove(keyAttribute)

> **Note**
>
> 明示的に指定しない属性は、上記の the section called “サポートされている 属性” の表に示したデフォルトに設定されます。

ビルダーパターンの例

通常、開発者にとっては、ビルダーパターンを介してクラスを利用する方がより便利です。例:

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;

CloudHsmKeyAttributesMap keyAttributesSessionDecryptionKey =
   new CloudHsmKeyAttributesMap.Builder()
      .put(CloudHsmKeyAttributes.CKA_LABEL, "ExtractableSessionKeyEncryptDecrypt")
      .put(CloudHsmKeyAttributes.CKA_WRAP, false)
```

```
      .put(CloudHsmKeyAttributes.CKA_UNWRAP, false)
      .put(CloudHsmKeyAttributes.CKA_SIGN, false)
      .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
      .build();

CloudHsmKeyAttributesMap keyAttributesTokenWrappingKey =
   new CloudHsmKeyAttributesMap.Builder()
      .put(CloudHsmKeyAttributes.CKA_LABEL, "TokenWrappingKey")
      .put(CloudHsmKeyAttributes.CKA_TOKEN, true)
      .put(CloudHsmKeyAttributes.CKA_ENCRYPT, false)
      .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
      .put(CloudHsmKeyAttributes.CKA_SIGN, false)
      .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
      .build();
```

開発者は、キーテンプレートのベストプラクティスを実施するための便利な方法として、事前に定義された属性セットを利用することもできます。例:

```
//best practice template for wrapping keys

CloudHsmKeyAttributesMap commonKeyAttrs = new CloudHsmKeyAttributesMap.Builder()
    .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, false)
    .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
    .build();

// initialize a new instance of CloudHsmKeyAttributesMap by copying commonKeyAttrs
// but with an appropriate label

CloudHsmKeyAttributesMap firstKeyAttrs = new CloudHsmKeyAttributesMap(commonKeyAttrs);
firstKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, "key label");

// alternatively, putAll() will overwrite existing values to enforce conformance

CloudHsmKeyAttributesMap secondKeyAttrs = new CloudHsmKeyAttributesMap();
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_DECRYPT, true);
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_ENCRYPT, true);
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, “safe wrapping key”);
secondKeyAttrs.putAll(commonKeyAttrs); // will overwrite CKA_DECRYPT to be FALSE
```

キーペアの属性設定

Java クラス CloudHsmKeyPairAttributesMap を使用して、キーペアのキー属性を処理します。CloudHsmKeyPairAttributesMap は、2 つの CloudHsmKeyAttributesMap オブジェクトをカプセル化します。1 つはパブリックキー用ともう 1 つはプライベートキー用です。

パブリックキーとプライベートキーの個々の属性を個別に設定するには、そのキーの対応する CloudHsmKeyAttributes マップオブジェクトで put() メソッドを使用できます。getPublic() メソッドを使用してパブリックキーの属性マップを取得し、getPrivate() を使用してプライベートキーの属性マップを取得します。引数としてキーペア属性マップを使用する putAll() を使用して、パブリックキーペアとプライベートキーペアの両方に、複数のキー属性の値を一緒に入力します。

ビルダーパターンの例

通常、開発者にとっては、ビルダーパターンを介してキー属性を設定する方がより便利です。例：

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;

//specify attributes up-front
CloudHsmKeyAttributesMap keyAttributes =
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_SIGN, false)
        .put(CloudHsmKeyAttributes.CKA_LABEL, "PublicCertSerial12345")
        .build();

CloudHsmKeyPairAttributesMap keyPairAttributes =
    new CloudHsmKeyPairAttributesMap.Builder()
        .withPublic(keyAttributes)
        .withPrivate(
            new CloudHsmKeyAttributesMap.Builder() //or specify them inline
                .put(CloudHsmKeyAttributes.CKA_LABEL, "PrivateCertSerial12345")
                .put (CloudHSMKeyAttributes.CKA_WRAP, FALSE)
                .build()
        )
        .build();
```

> **Note**
>
> この独自の拡張機能の詳細については、GitHub の Javadoc アーカイブと sample を参照してください。Javadoc を調べるには、アーカイブをダウンロードして展開します。

まとめ

キーオペレーションでキー属性を指定するには、次の手順に従います。

1. 対称キーの CloudHsmKeyAttributesMap、またはキーペアの CloudHsmKeyPairAttributesMap をインスタンス化します。
2. 必要なキー属性と値を使用して、ステップ 1 からの属性オブジェクトを定義します。
3. 特定のキータイプに対応する Cavium*ParameterSpec クラスをインスタンス化し、この設定された属性オブジェクトをコンストラクタに渡します。
4. この Cavium*ParameterSpec オブジェクトを対応する暗号クラスまたはメソッドに渡します。

参考のために、次の表に、カスタムキー属性をサポートする Cavium*ParameterSpec クラスとメソッドを示します。

キータイプ	パラメータ仕様クラス	コンストラクタの例
基本クラス	`CaviumKeyGenAlgorithmParameterSpec`	`CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)`
DES	`CaviumDESKeyGenParameterSpec`	`CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)`

キータイプ	パラメータ仕様クラス	コンストラクタの例
RSA	CaviumRSAKeyGenParameterSpec	CaviumRSAKeyGenParameterSpec(int keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)
シークレット	CaviumGenericSecretKeyGenParameterSpec	CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttributesMap key AttributesMap)
AES	CaviumAESKeyGenParameterSpec	CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap key AttributesMap)
EC	CaviumECGenParameterSpec	CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)

サンプルコード: キーの生成とラップ

次の簡単なコードサンプルは、キー生成とキーラップの 2 つの異なるオペレーションの手順を示しています。

```
// Set up the desired key attributes
```

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec keyAttributes = new CaviumAESKeyGenParameterSpec(
    256,
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_LABEL, "MyPersistentAESKey")
        .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, true)
        .put(CloudHsmKeyAttributes.CKA_TOKEN, true)
        .build()
);

// Assume we already have a handle to the myWrappingKey
// Assume we already have the wrappedBytes to unwrap

// Unwrap a key using Custom Key Attributes

CaviumUnwrapParameterSpec unwrapSpec = new
 CaviumUnwrapParameterSpec(myInitializationVector, keyAttributes);

Cipher unwrapCipher = Cipher.getInstance("AESWrap", "Cavium");
unwrapCipher.init(Cipher.UNWRAP_MODE, myWrappingKey, unwrapSpec);
Key unwrappedKey = unwrapCipher.unwrap(wrappedBytes, "AES", Cipher.SECRET_KEY);
```

Java for Client SDK 3 用の AWS CloudHSM ソフトウェアライブラリのコードサンプル

このトピックでは、 AWS CloudHSM クライアント SDK 3 の Java コードサンプルに関するリソースと情報を提供します。

前提条件

サンプルを実行する前に、環境をセットアップする必要があります。

- Java Cryptographic Extension (JCE) provider と AWS CloudHSM client package をインストールします。
- 有効な HSM ユーザー名とパスワードを設定します。これらのタスクには、暗号化ユーザー (CU) のアクセス権限で十分です。アプリケーションは、それぞれの例でこの認証情報を使用して HSM にログインします。
- JCE provider へのクレデンシャルを提供する方法を決定します。

コードサンプル

次のコードサンプルでは、基本タスクを実行するために、AWS CloudHSM JCE provider を使用する方法を示します。その他の例は GitHub から入手できます。

- HSM へのログイン
- キーの管理
- AES キーの生成
- AES GCM による暗号化と復号
- Encrypt and decrypt with AES-CTR
- D3DES-ECB による暗号化と復号[注記 1 参照]
- AES-GCM を使用したキーのラップとラップ解除
- AES を使用したキーのラップとラップ解除
- RSA を使用したキーのラップとラップ解除
- サポートされているキー属性の使用
- キーストアのキーの列挙
- CloudHSM キーストアの使用
- マルチスレッドでのメッセージの署名のサンプル
- Sign and Verify with EC Keys

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

クライアント SDK 3 のAWS CloudHSM KeyStore Java クラス

クラスは、keytool AWS CloudHSM `KeyStore`や jarsigner などのアプリケーションを通じて AWS CloudHSM キーへのアクセスを許可する専用 PKCS12 キーストアを提供します。このキーストアでは、証明書をキーデータとともに保存し、 AWS CloudHSMに保存されているキーデータに関連付けることができます。

> **Note**
>
> 証明書は公開情報であり、暗号化キーのストレージ容量を最大化するため、 AWS CloudHSM は HSMs への証明書の保存をサポートしていません。

クラスは、Java AWS CloudHSM KeyStoreCryptography Extension (JCE) のKeyStoreサービスプロバイダーインターフェイス (SPI) を実装します。KeyStore の使用の詳細については、「Class KeyStore」を参照してください。

AWS CloudHSM クライアント SDK 3 に適したキーストアを選択する

AWS CloudHSM Java 暗号化拡張 (JCE) プロバイダーには、すべてのトランザクションを HSM に渡すデフォルトのパススルー読み取り専用キーストアが付属しています。このデフォルトのキーストアは、特殊な目的の AWS CloudHSM KeyStore とは異なります。ほとんどの場合、デフォルトを使用することにより、ランタイムのパフォーマンスとスループットが向上します。 AWS CloudHSM KeyStoreは、キーオペレーションを HSM にオフロードするだけでなく、証明書と証明書ベースのオペレーションのサポートが必要なアプリケーションにのみ使用してください。

どちらのキーストアも操作に Cavium JCE プロバイダを使用しますが、これらは独立したエンティティであり、相互に情報を交換しません。

Java アプリケーションのデフォルトのキーストアを次のようにロードします。

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

次のように、特殊目的の CloudHSM KeyStore をロードします。

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

クライアント SDK 3 の AWS CloudHSM KeyStore を初期化する

JCE プロバイダーにログインするのと同じ方法で、 AWS CloudHSM KeyStore にログインします。環境変数またはシステムプロパティファイルを使用できます。CloudHSM KeyStore を使用する前にログインする必要があります。JCE プロバイダーを使用して HSM にログインする例については、Login to an HSM を参照してください。

必要に応じて、パスワードを指定して、キーストアデータを保持するローカル PKCS12 ファイルを暗号化できます。 AWS CloudHSM Keystore を作成するときは、パスワードを設定し、ロード、設定、取得の方法を使用するときに指定します。

新しい CloudHSM KeyStore オブジェクトを次のようにインスタンス化します。

```
ks.load(null, null);
```

store メソッドを使用して、キーストアデータをファイルに書き込みます。その後は、次のように、ソースファイルとパスワードを使用し、load メソッドを使用して既存のキーストアをロードできます。

```
ks.load(inputStream, password);
```

AWS CloudHSM クライアント SDK 3 に AWS CloudHSM KeyStoreを使用する

CloudHSM KeyStore オブジェクトは、通常、jarsigner や keytool などのサードパーティー製アプリケーションを通じて使用されます。コードを使用してオブジェクトに直接アクセスすることもできます。

AWS CloudHSM KeyStore は JCE Class KeyStore 仕様に準拠しており、次の関数を提供します。

- load

  指定された入力ストリームからキーストアをロードします。キーストアの保存時にパスワードが設定されている場合、ロードを成功させるには、この同じパスワードを指定する必要があります。新しい空のキーストアを初期化するには、両方のパラメータを null に設定します。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- aliases

  指定されたキーストアインスタンス内に含まれるすべてのエントリのエイリアス名の列挙を返します。結果には、PKCS12 ファイルにローカルに保存されたオブジェクトと、HSM 上に存在するオブジェクトが含まれます。

  サンプルコード:

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();)
   {
    String label = entry.nextElement();
    System.out.println(label);
   }
  ```

- ContainsAlias

キーストアが、指定されたエイリアスを持つ少なくとも 1 つのオブジェクトにアクセスできる場合は true を返します。キーストアは、PKCS12 ファイルにローカルに保存されているオブジェクトと、HSM 上に存在するオブジェクトをチェックします。

- `DeleteEntry`

  ローカル PKCS12 ファイルから証明書エントリを削除します。HSM に保存されているキーデータの削除は、 AWS CloudHSM KeyStore ではサポートされていません。CloudHSM の key_mgmt_util ツールを使用してキーを削除できます。

- `GetCertificate`

  使用可能な場合、エイリアスに関連付けられた証明書を返します。エイリアスが存在しないか、証明書ではないオブジェクトを参照している場合、関数は NULL を返します。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias)
```

- `GetCertificateAlias`

  指定された証明書とデータが一致する最初のキーストアエントリの名前 (エイリアス) を返します。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert)
```

- `GetCertificateChain`

  指定されたエイリアスに関連付けられた証明書チェーンを返します。エイリアスが存在しないか、証明書ではないオブジェクトを参照している場合、関数は NULL を返します。

- `GetCreationDate`

  指定されたエイリアスによって識別されるエントリの作成日を返します。作成日が使用できない場合、この関数は証明書が有効になった日付を返します。

- `GetKey`

  GetKey が HSM に渡され、指定されたラベルに対応するキーオブジェクトを返します。`getKey` が HSM を直接照会すると、KeyStore によって生成されたかどうかに関係なく、HSM 上の任意のキーに使用できます。

```
Key key = ks.getKey(keyLabel, null);
```

- IsCertificateEntry

  指定されたエイリアスを持つエントリが証明書エントリを表すかどうかをチェックします。

- IsKeyEntry

  指定されたエイリアスを持つエントリがキーエントリを表すかどうかをチェックします。このアクションは、PKCS12 ファイルと HSM の両方でエイリアスを検索します。

- SetCertificateEntry

  指定された証明書を指定されたエイリアスに割り当てます。指定されたエイリアスがキーまたは証明書の識別にすでに使用されている場合は、KeyStoreException がスローされます。JCE コードを使用してキーオブジェクトを取得し、KeyStore SetKeyEntry メソッドを使用して証明書をキーに関連付けることができます。

- byte[] キーのある SetKeyEntry

  この API は現在、クライアント SDK 3 ではサポートされていません。

- Key オブジェクトのある SetKeyEntry

  指定されたキーを指定されたエイリアスに割り当て、HSM 内に保存します。Key オブジェクトが CaviumKey のタイプでない場合、キーは抽出可能なセッションキーとして HSM にインポートされます。

  Key オブジェクトが PrivateKey のタイプの場合、対応する証明書チェーンが添付されている必要があります。

  エイリアスが既に存在する場合、SetKeyEntry 呼び出しは KeyStoreException をスローし、キーが上書きされるのを防ぎます。キーを上書きする必要がある場合は、そのために KMU または JCE を使用します。

- EngineSize

  キーストア内のエントリの数を返します。

- Store

  キーストアを指定された出力ストリームに PKCS12 ファイルとして保存し、指定されたパスワードで保護します。さらに、ロードされたすべてのキー (setKey 呼び出しを使用して設定される) が保持されます。

暗号化 API: の次世代 (CNG) およびキーストレージプロバイダー (KSP) AWS CloudHSM

Windows 用の AWS CloudHSM クライアントには、CNG プロバイダーと KSP プロバイダーが含まれています。

Key storage providers (KSPs)により、キーの格納と取得が可能になります。たとえば、Microsoft の Active Directory Certificate Services (AD CS) の役割を Windows サーバーに追加し、認証局 (CA) の新しいプライベートキーを作成するのを選択した場合は、キーストレージを管理する KSP を選択できます。AD CS のロールを設定するときは、KSP を選択できます。詳細については、「Windows Server CA の作成」を参照してください。

Cryptography API: Next Generation (CNG) は、Microsoft Windows オペレーティングシステム固有の暗号化 API です。CNG を使用すると、開発者は暗号化技術を使用して Windows ベースのアプリケーションを保護できます。大まかに言うと、CNG の AWS CloudHSM 実装には以下の機能があります。

- 暗号化プリミティブ - 基本的な暗号化オペレーションを実行することができます。
- Key Import and Export - 対称キーと非対称キーのインポートとエクスポートを可能にします。
- Data Protection API (CNG DPAPI) - データの暗号化と復号化を簡単にすることができます。
- キーの保管と取り出し - 非対称キーペアのプライベートキーを安全に保管し隔離することができます。

トピック

- の KSP プロバイダーと CNG プロバイダーを検証する AWS CloudHSM
- AWS CloudHSM Windows クライアントを使用するための前提条件
- AWS CloudHSM キーを証明書に関連付ける
- の CNG プロバイダーのコードサンプル AWS CloudHSM

の KSP プロバイダーと CNG プロバイダーを検証する AWS CloudHSM

KSP および CNG プロバイダーは、Windows AWS CloudHSM クライアントのインストール時にインストールされます。クライアントは、「クライアントのインストール (Windows)」の手順に従ってインストールします。

以下のセクションを使用して、プロバイダーのインストールを確認します。

Windows AWS CloudHSM クライアントを設定して実行する

Windows CloudHSM クライアントを開始する前に、前提条件 を満たす必要があります。次に、プロバイダーが使用する設定ファイルを更新し、以下の手順を実行してクライアントを起動します。これらのステップは、KSP および CNG プロバイダーの初回使用時と、クラスターの HSM の追加または削除を行った後に必要です。これにより、クラスター内のすべての HSM AWS CloudHSM 間でデータを同期し、一貫性を維持できます。 HSMs

ステップ 1: AWS CloudHSM クライアントを停止する

プロバイダーが使用する設定ファイルを更新する前に、 AWS CloudHSM クライアントを停止します。クライアントが停止済みである場合、stop コマンドを実行しても影響はありません。

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合。

  AWS CloudHSM クライアントを起動したコマンドウィンドウで Ctrl+C を使用します。

ステップ 2: AWS CloudHSM 設定ファイルを更新する

この手順では、-aConfigure ツールの パラメータを使用して、クラスター内の HSM の 1 つの Elastic Network Interface (ENI) IP アドレスを設定ファイルに追加します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM ENI IP>
```

クラスター内の HSM の ENI IP アドレスを取得するには、 AWS CloudHSM コンソールに移動し、クラスターを選択し、目的のクラスターを選択します。DescribeClusters オペレーション、describe-clusters コマンド、または Get-HSM2Cluster PowerShell コマンドレットを使用することもできます。1 つの ENI IP アドレスのみを入力します。どの ENI IP アドレスでも使用できます。

ステップ 3: AWS CloudHSM クライアントを起動する

次に、 AWS CloudHSM クライアントを起動または再起動します。 AWS CloudHSM クライアントは起動時に、設定ファイルの ENI IP アドレスを使用してクラスターをクエリします。次に、クラスター内のすべての HSM の ENI IP アドレスを、クラスター情報ファイルに追加します。

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

KSP および CNG プロバイダーの確認

次のいずれかのコマンドを使用して、システムにインストールするプロバイダーを決定します。コマンドは、登録された KSP および CNG プロバイダーをリスト表示。 AWS CloudHSM クライアントを実行する必要はありません。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\ksp_config.exe" -enum
```

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cng_config.exe" -enum
```

KSP および CNG プロバイダーが Windows Server EC2 インスタンスにインストールされていることを確認するには、リスト中に次のエントリが表示されているのを見る必要があります。

```
Cavium CNG Provider
Cavium Key Storage Provider
```

CNG プロバイダーが見つからない場合は、次のコマンドを実行します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cng_config.exe" -register
```

CNG プロバイダーが見つからない場合は、次のコマンドを実行します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\ksp_config.exe" -register
```

AWS CloudHSM Windows クライアントを使用するための前提条件

Windows AWS CloudHSM クライアントを起動して KSP および CNG プロバイダーを使用する前に、システムで HSM のログイン認証情報を設定する必要があります。Windows 資格情報マネージャーまたはシステム環境変数を使用して、認証情報を設定できます。認証情報の保存に

は、Windows 資格情報マネージャーを使用することをお勧めします。このオプションは、 AWS CloudHSM クライアントバージョン 2.0.4 以降で使用できます。環境変数を使用すると設定が簡単になりますが、Windows 資格情報マネージャーを使用するよりも安全性が低くなります。

Windows 資格情報マネージャー

set_cloudhsm_credentials ユーティリティまたは Windows 資格情報マネージャーのインターフェイスのいずれかを使用できます。

- **set_cloudhsm_credentials** ユーティリティの使用:

  set_cloudhsm_credentials ユーティリティは Windows インストーラに含まれています。このユーティリティを使用して、HSM ログイン認証情報を Windows 資格情報マネージャーに簡単に渡すことができます。このユーティリティをソースからコンパイルする場合は、インストーラに含まれている Python コードを使用できます。

  1. C:\Program Files\Amazon\CloudHSM\tools\ フォルダに移動します。
  2. CU ユーザー名とパスワードのパラメータを使用して set_cloudhsm_credentials.exe ファイルを実行します。

     ```
     set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
     ```

- 資格情報マネージャーのインターフェイスの使用:

  資格情報マネージャーのインターフェイスを使用して、認証情報を手動で管理できます。

  1. 資格情報マネージャーを開くには、タスクバーの検索ボックスに「credential manager」と入力し、[資格情報マネージャー] を選択します。
  2. [Windows 資格情報] を選択して、Windows 資格情報を管理します。
  3. [汎用資格情報の追加] を選択し、以下のように詳細を入力します。
     - [インターネットまたはネットワークアドレス] にターゲット名として「cloudhsm_client」と入力します。
     - [ユーザー名] と [パスワード] に CU 認証情報を入力します。
     - [OK] をクリックします。

システム環境変数

Windows アプリケーションの HSM および Crypto User (CU) を識別するシステム環境変数を設定できます。setx コマンドを使用して、システム環境変数を設定するか、permanent システム環境変数

を programmatically に設定するか、あるいは Windows の System Properties コントロールパネルの Advanced タブ中に設定します。

> **Warning**
>
> システム環境変数を使用して認証情報を設定すると、ユーザーのシステムでパスワードがプレーンテキストで入手可能になります。この問題を解決するには、Windows 資格情報マネージャーを使用します。

次のシステム環境変数を設定します。

n3fips_password=*<CU USERNAME>*:*<CU PASSWORD>*

HSM の Crypto User (CU) を識別し、必要なすべてのログイン情報を提供します。アプリケーションはこの CU として認証および実行します。このアプリケーションには、この CU のアクセス権限があり、CU が所有および共有しているキーのみを表示および管理できます。新しい CU を作成するには、createUser を使用します。既存の CU を検索するには、listUsers を使用します。

例：

```
setx /m n3fips_password test_user:password123
```

AWS CloudHSM キーを証明書に関連付ける

Microsoft の SignTool などのサードパーティーツールで AWS CloudHSM キーを使用する前に、キーのメタデータをローカル証明書ストアにインポートし、メタデータを証明書に関連付ける必要があります。キーのメタデータをインポートするには、CloudHSM バージョン 3.0 以降に含まれている import_key.exe ユーティリティを使用します。次の手順では、追加情報とサンプル出力を示します。

ステップ 1: 証明書をインポートする

Windows では、証明書をダブルクリックするとローカルの証明書ストアにインポートできます。

ただし、ダブルクリックしてもインポートできない場合は、Microsoft Certreq ツールを使用して証明書マネージャーに証明書をインポートします。例えば:

```
certreq -accept <certificatename>
```

この操作が失敗し、エラー Key not found が表示された場合は、手順 2 に進みます。証明書がキーストアに表示される場合は、タスクは完了しているため、これ以上の操作は必要ありません。

ステップ 2: 証明書識別情報を収集する

前の手順が成功しなかった場合は、プライベートキーを証明書に関連付ける必要があります。ただし、関連付けを作成する前に、まず証明書の一意のコンテナ名とシリアル番号を検索する必要があります。certutil などのユーティリティを使用して、必要な証明書情報を表示します。certutil からの次の出力例は、コンテナ名とシリアル番号を示しています。

```
================ Certificate 1 ================ Serial Number:
   72000000047f7f7a9d41851b4e000000000004Issuer: CN=Enterprise-CANotBefore: 10/8/2019
 11:50
   AM NotAfter: 11/8/2020 12:00 PMSubject: CN=www.example.com, OU=Certificate
 Management,
   O=Information Technology, L=Seattle, S=Washington, C=USNon-root CertificateCert
   Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45 75 bc 65No key
 provider
   information Simple container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
 Unique
   container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
```

ステップ 3: AWS CloudHSM プライベートキーを証明書に関連付ける

キーを証明書に関連付けるには、まずAWS CloudHSM クライアントデーモンを起動してください。次に、import_key.exe (CloudHSM バージョン 3.0 以降に含まれています) を使用して、プライベートキーを証明書に関連付けます。証明書を指定するときは、その単純なコンテナ名を使用します。次の例は、コマンドと応答を示しています。このアクションでは、キーのメタデータのみがコピーされます。キーは HSM に残ります。

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c

Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
 80090016
```

ステップ 4: 証明書ストアを更新する

AWS CloudHSM クライアントデーモンがまだ実行されていることを確認します。次に、certutil 動詞の -repairstore を使用して、証明書のシリアル番号を更新します。次のサンプルは、コマンドと出力の例を示しています。-repairstore 動詞の詳細については、Microsoft のドキュメントを参照してください。

```
C:\Program Files\Amazon\CloudHSM>certutil -f -csp "Cavium Key Storage Provider"-
repairstore my "72000000047f7f7a9d41851b4e000000000004"
my "Personal"
================ Certificate 1 ================
Serial Number: 72000000047f7f7a9d41851b4e000000000004
Issuer: CN=Enterprise-CA
NotBefore: 10/8/2019 11:50 AM
NotAfter: 11/8/2020 12:00 PM
Subject: CN=www.example.com, OU=Certificate Management, O=Information Technology,
 L=Seattle, S=Washington, C=US
Non-root CertificateCert Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45
 75 bc 65
SDK Version: 3.0
Key Container = CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
Provider = "Cavium Key Storage Provider"
Private key is NOT exportableEncryption test passedCertUtil: -repairstore command
 completed successfully.
```

証明書のシリアル番号を更新したら、この証明書と対応する AWS CloudHSM プライベートキーを Windows の任意のサードパーティー署名ツールで使用できます。

の CNG プロバイダーのコードサンプル AWS CloudHSM

⚠ ** コード例のみ - 本稼働環境で使用しないでください **
このサンプルコードは、例示のみを目的としています。本稼働環境でこのコードを実行しないでください。

次の例では、Windows のための CloudHSM を用いてインストールされた CNG プロバイダーを見つけるために、登録された暗号化プロバイダーをシステム上で列挙する方法について示しています。この例では、非対称キーペアの作成方法と、キーペアを使用してデータに署名する方法も示しています。

⚠ Important

この例を実行する前に、前提条件の説明に従って HSM 認証情報を設定する必要があります。詳細については、AWS CloudHSM Windows クライアントを使用するための前提条件 を参照してください

```
// CloudHsmCngExampleConsole.cpp : Console application that demonstrates CNG
 capabilities.
// This example contains the following functions.
//
//   VerifyProvider()          - Enumerate the registered providers and retrieve Cavium
 KSP and CNG providers.
//   GenerateKeyPair()         - Create an RSA key pair.
//   SignData()                - Sign and verify data.
//

#include "stdafx.h"
#include <Windows.h>

#ifndef NT_SUCCESS
#define NT_SUCCESS(Status) ((NTSTATUS)(Status) >= 0)
#endif

#define CAVIUM_CNG_PROVIDER L"Cavium CNG Provider"
#define CAVIUM_KEYSTORE_PROVIDER L"Cavium Key Storage Provider"

// Enumerate the registered providers and determine whether the Cavium CNG provider
// and the Cavium KSP provider exist.
//
bool VerifyProvider()
{
  NTSTATUS status;
  ULONG cbBuffer = 0;
  PCRYPT_PROVIDERS pBuffer = NULL;
  bool foundCng = false;
  bool foundKeystore = false;

  // Retrieve information about the registered providers.
  //   cbBuffer - the size, in bytes, of the buffer pointed to by pBuffer.
  //   pBuffer - pointer to a buffer that contains a CRYPT_PROVIDERS structure.
  status = BCryptEnumRegisteredProviders(&cbBuffer, &pBuffer);

  // If registered providers exist, enumerate them and determine whether the
  // Cavium CNG provider and Cavium KSP provider have been registered.
  if (NT_SUCCESS(status))
  {
    if (pBuffer != NULL)
```

```
    {
      for (ULONG i = 0; i < pBuffer->cProviders; i++)
      {
        // Determine whether the Cavium CNG provider exists.
        if (wcscmp(CAVIUM_CNG_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
        {
          printf("Found %S\n", CAVIUM_CNG_PROVIDER);
          foundCng = true;
        }

        // Determine whether the Cavium KSP provider exists.
        else if (wcscmp(CAVIUM_KEYSTORE_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
        {
          printf("Found %S\n", CAVIUM_KEYSTORE_PROVIDER);
          foundKeystore = true;
        }
      }
    }
  }
  else
  {
    printf("BCryptEnumRegisteredProviders failed with error code 0x%08x\n", status);
  }

  // Free memory allocated for the CRYPT_PROVIDERS structure.
  if (NULL != pBuffer)
  {
    BCryptFreeBuffer(pBuffer);
  }

  return foundCng == foundKeystore == true;
}

// Generate an asymmetric key pair. As used here, this example generates an RSA key
 pair
// and returns a handle. The handle is used in subsequent operations that use the key
 pair.
// The key material is not available.
//
// The key pair is used in the SignData function.
//
NTSTATUS GenerateKeyPair(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE *hKey)
{
  NTSTATUS status;
```

```
  // Generate the key pair.
  status = BCryptGenerateKeyPair(hAlgorithm, hKey, 2048, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptGenerateKeyPair failed with code 0x%08x\n", status);
    return status;
  }

  // Finalize the key pair. The public/private key pair cannot be used until this
  // function is called.
  status = BCryptFinalizeKeyPair(*hKey, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptFinalizeKeyPair failed with code 0x%08x\n", status);
    return status;
  }

  return status;
}

// Sign and verify data using the RSA key pair. The data in this function is hardcoded
// and is for example purposes only.
//
NTSTATUS SignData(BCRYPT_KEY_HANDLE hKey)
{
  NTSTATUS status;
  PBYTE sig;
  ULONG sigLen;
  ULONG resLen;
  BCRYPT_PKCS1_PADDING_INFO pInfo;

  // Hardcode the data to be signed (for demonstration purposes only).
  PBYTE message = (PBYTE)"d83e7716bed8a20343d8dc6845e57447";
  ULONG messageLen = strlen((char*)message);

  // Retrieve the size of the buffer needed for the signature.
  status = BCryptSignHash(hKey, NULL, message, messageLen, NULL, 0, &sigLen, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptSignHash failed with code 0x%08x\n", status);
    return status;
  }
```

```
  // Allocate a buffer for the signature.
  sig = (PBYTE)HeapAlloc(GetProcessHeap(), 0, sigLen);
  if (sig == NULL)
  {
    return -1;
  }

  // Use the SHA256 algorithm to create padding information.
  pInfo.pszAlgId = BCRYPT_SHA256_ALGORITHM;

  // Create a signature.
  status = BCryptSignHash(hKey, &pInfo, message, messageLen, sig, sigLen, &resLen,
 BCRYPT_PAD_PKCS1);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptSignHash failed with code 0x%08x\n", status);
    return status;
  }

  // Verify the signature.
  status = BCryptVerifySignature(hKey, &pInfo, message, messageLen, sig, sigLen,
 BCRYPT_PAD_PKCS1);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptVerifySignature failed with code 0x%08x\n", status);
    return status;
  }

  // Free the memory allocated for the signature.
  if (sig != NULL)
  {
    HeapFree(GetProcessHeap(), 0, sig);
    sig = NULL;
  }

  return 0;
}

// Main function.
//
int main()
{
  NTSTATUS status;
  BCRYPT_ALG_HANDLE hRsaAlg;
```

```
  BCRYPT_KEY_HANDLE hKey = NULL;

  // Enumerate the registered providers.
  printf("Searching for Cavium providers...\n");
  if (VerifyProvider() == false) {
    printf("Could not find the CNG and Keystore providers\n");
    return 1;
  }

  // Get the RSA algorithm provider from the Cavium CNG provider.
  printf("Opening RSA algorithm\n");
  status = BCryptOpenAlgorithmProvider(&hRsaAlg, BCRYPT_RSA_ALGORITHM,
CAVIUM_CNG_PROVIDER, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptOpenAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
  }

  // Generate an asymmetric key pair using the RSA algorithm.
  printf("Generating RSA Keypair\n");
  GenerateKeyPair(hRsaAlg, &hKey);
  if (hKey == NULL)
  {
    printf("Invalid key handle returned\n");
    return 0;
  }
  printf("Done!\n");

  // Sign and verify [hardcoded] data using the RSA key pair.
  printf("Sign/Verify data with key\n");
  SignData(hKey);
  printf("Done!\n");

  // Remove the key handle from memory.
  status = BCryptDestroyKey(hKey);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptDestroyKey failed with code 0x%08x\n", status);
    return status;
  }

  // Close the RSA algorithm provider.
  status = BCryptCloseAlgorithmProvider(hRsaAlg, NULL);
```

```
  if (!NT_SUCCESS(status))
  {
    printf("BCryptCloseAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
  }

  return 0;
}
```

サードパーティーアプリケーションを と統合する AWS CloudHSM

のユースケースには、サードパーティーのソフトウェアアプリケーションを AWS CloudHSM クラスター内の HSM と統合すること AWS CloudHSM が含まれます。サードパーティーのソフトウェアを と統合することで AWS CloudHSM、さまざまなセキュリティ関連の目標を達成できます。以下のトピックでは、これらの目標のいくつかを達成する方法について説明します。

トピック

- での SSL/TLS オフロードによるウェブサーバーセキュリティの向上 AWS CloudHSM
- で認証機関 (CA) として Windows Server を設定する AWS CloudHSM
- を使用した Oracle データベース透過的なデータ暗号化 (TDE) AWS CloudHSM
- で Microsoft SignTool を使用してファイルに署名 AWS CloudHSM する
- Java Keytool と Jarsigner の統合 AWS CloudHSM
- で Microsoft Manifest Generation and Editing Tool (Mage.exe) AWS CloudHSM を使用してファイルに署名する
- とのその他のサードパーティーベンダー統合 AWS CloudHSM

での SSL/TLS オフロードによるウェブサーバーセキュリティの向上 AWS CloudHSM

ウェブサーバーとそのクライアント (ウェブブラウザ) では、Secure Sockets Layer (SSL) または Transport Layer Security (TLS) プロトコルを使用して、ウェブサーバーのアイデンティティを確認し、インターネット上でウェブページやその他のデータを送受信するための安全な接続を確立するための安全な接続を確立するものです。これは HTTPS として知られています。このウェブサーバーでは、パブリック/プライベートのキーペアと SSL/TLS パブリックキー証明書を使用して、各クライアントとの HTTPS セッションを確立します。このプロセスにはウェブサーバーの多くの計算が含まれますが、その一部を SSL アクセラレーションと呼ばれる AWS CloudHSM クラスターにオフロードできます。オフロードすることで、ウェブサーバーの計算負荷が軽減され、サーバーのプライベートキーを HSM に保存することでセキュリティが強化されます。

以下のトピックでは、 での SSL/TLS オフロードの AWS CloudHSM 仕組みの概要と、以下のプラットフォーム AWS CloudHSM で で SSL/TLS オフロードを設定するためのチュートリアルについて説明します。

Linux の場合は、NGINX または Apache HTTP Server Web サーバーソフトウェアで OpenSSL Dynamic Engine を使用します

Windows の場合、Windows Server の Internet Information Services (IIS) のウェブサーバーソフトウェアを使用します

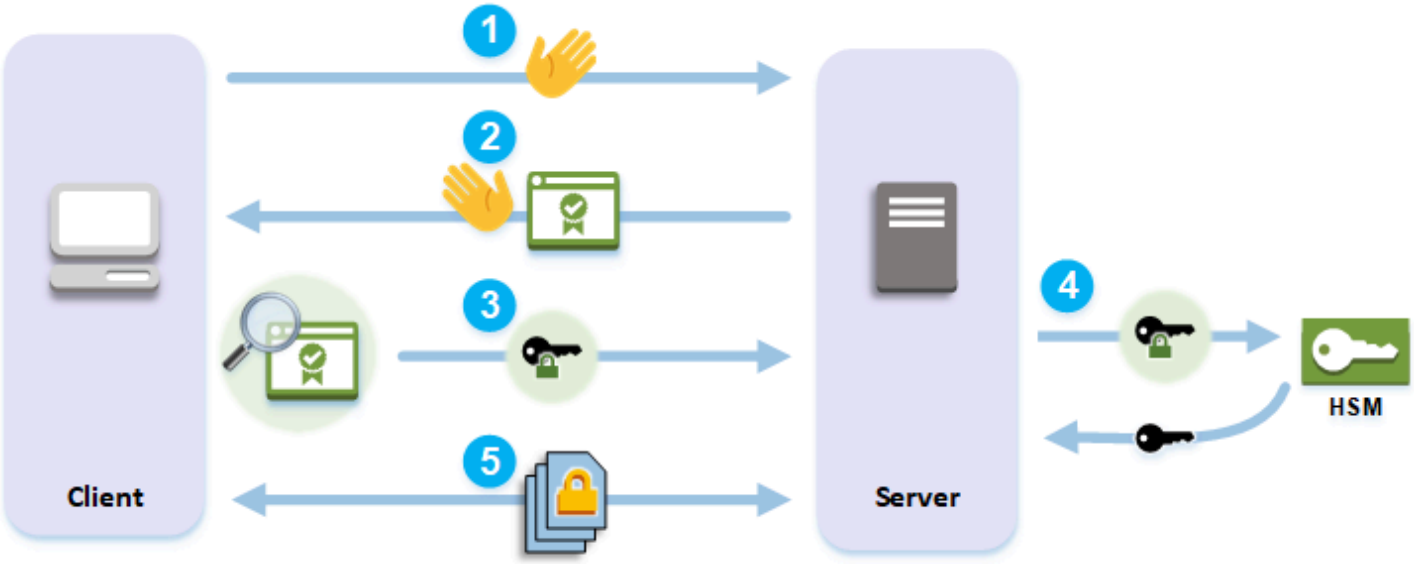
トピック

- での SSL/TLS オフロードの AWS CloudHSM 仕組み
- AWS CloudHSM NGINX または Apache with OpenSSL を使用した Linux での SSL/TLS オフロード
- AWS CloudHSM OpenSSL プロバイダーで NGINX または HAProxy を使用した Linux での OpenSSL/TLS オフロード
- AWS CloudHSM JSSE で Tomcat を使用した Linux での SSL/TLS オフロード
- AWS CloudHSM KSP で IIS を使用した Windows での SSL/TLS オフロード
- Elastic Load Balancing for でロードバランサーを追加する AWS CloudHSM(オプション)

での SSL/TLS オフロードの AWS CloudHSM 仕組み

HTTPS 接続を確立するため、ウェブサーバーはクライアントとハンドシェークプロセスを実行します。このプロセスの一環として、次の図に示すように、サーバーは暗号化処理の一部を AWS CloudHSM クラスター内の HSMs にオフロードします。プロセスの各ステップについて、図の下に説明があります。

> **Note**
>
> 次のイメージとプロセスでは、サーバーの検証とキーの交換に RSA を使用することを想定しています。RSA の代わりに Diffie-Hellman を使用する場合、手順が若干異なります。



1. クライアントはサーバーに Hello メッセージを送信します。
2. サーバーは Hello メッセージで応答し、サーバーの証明書を送信します。
3. クライアントは以下のアクションを実行します。
   a. SSL/TLS サーバーの証明書がクライアントの信頼するルート証明書により署名されていることを確認します。
   b. サーバーの証明書からパブリックキーを抽出します。
   c. プリマスターシークレットを生成し、サーバーのパブリックキーで暗号化します。
   d. 暗号化されたプリマスターシークレットをサーバーに送信します。
4. クライアントのプリマスターシークレットを復号するため、サーバーはそれを HSM に送信します。HSM は、HSM のプライベートキーを使用してプリマスターシークレットを復号し、サーバーに送信します。クライアントとサーバーはそれぞれ独立してプリマスターシークレットおよび Hello メッセージからのいくつかの情報を使用して、マスターシークレットを計算します。
5. ハンドシェイクプロセスは終了します。残りのセッションでは、クライアントとサーバーの間で送信されたすべてのメッセージは、マスターシークレットのデリバティブで暗号化されます。

で SSL/TLS オフロードを設定する方法については AWS CloudHSM、次のいずれかのトピックを参照してください。

- AWS CloudHSM NGINX または Apache with OpenSSL を使用した Linux での SSL/TLS オフロード
- AWS CloudHSM JSSE で Tomcat を使用した Linux での SSL/TLS オフロード

- AWS CloudHSM KSP で IIS を使用した Windows での SSL/TLS オフロード

AWS CloudHSM NGINX または Apache with OpenSSL を使用した Linux での SSL/TLS オフロード

このトピックでは、Linux ウェブサーバー上の AWS CloudHSM を使用して SSL/TLS オフロードを設定する手順を説明します。

トピック

- 概要
- ステップ 1: 前提条件の設定
- ステップ 2: プライベートキーと SSL/TLS 証明書を生成する
- ステップ 3: ウェブサーバーを設定する
- ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

概要

Linux では、NGINX と Apache HTTP サーバーのウェブサーバーソフトウェアは OpenSSL とネイティブに統合して HTTPS をサポートします。OpenSSL 用AWS CloudHSM 動的エンジンには、暗号化オフロードとキーストレージ用に、ウェブサーバーソフトウェアがクラスターの HSM を使用することを許可するインターフェイスが用意されています。OpenSSL エンジンは、ウェブサーバーと AWS CloudHSM クラスターの橋渡しをします。

このチュートリアルを完了するには、まず、Linux 上で NGINX と Apache のどちらのウェブサーバーを使用するかを選択する必要があります。選択したら、チュートリアルに以下の方法が表示されます。

- Amazon EC2 インスタンスに、ウェブサーバーソフトウェアをインストールします。
- AWS CloudHSM クラスターに保存されている秘密キーで、HTTPS をサポートするようにウェブサーバソフトウェアを設定します。
- (オプション) Amazon EC2 を使用して 2 台目のウェブサーバーインスタンスを作成し、Elastic Load Balancing を使用してロードバランサーを作成します。ロードバランサーを使用すると、複数のサーバーに負荷を分散することでパフォーマンスを向上させることができます。また、1 つ以上のサーバーに障害が発生した場合、冗長性と高可用性を提供します。

始める準備ができたら、「ステップ 1: 前提条件の設定」を参照してください。

ステップ 1: 前提条件の設定

プラットフォームごとに、異なる前提条件が必要です。以下の前提条件セクションのうち、お使いのプラットフォームに合ったものをご利用ください。

クライアント SDK 5 の前提条件

クライアント SDK 5でウェブサーバー SSL/TLS オフロードを設定するには、以下が必要です。

- 少なくとも 2 つのハードウェアセキュリティモジュール (HSM) を持つアクティブな AWS CloudHSM クラスター

  > **Note**
  >
  > HSM クラスターは1つでも使用できますが、まずクライアントキーの耐久性を無効にする必要があります。詳細については、クライアントキーの耐久性設定の管理 そして クライアント SDK 5 設定ツール を参照してください。

- Amazon EC2 インスタンスが Linux オペレーティングシステムを実行します。インスタンスに次のソフトウェアがインストールされていることを確認します。
  - ウェブサーバー (NGINX または Apache)
  - クライアント SDK 5 の OpenSSL 動的エンジン
- HSM でこのウェブサーバーのプライベートキーを所有および管理する Crypto User (CU)。

Linux ウェブサーバーインスタンスをセットアップし、HSM で CU を作成するには

1. OpenSSL Dynamic Engine をインストールして設定します AWS CloudHSM。OpenSSL ダイナミックエンジンのインストールの詳細については、クライアント SDK 5 の OpenSSL 動的エンジン を参照してください。
2. クラスターにアクセスできる EC2 Linux インスタンスで、NGINX または Apache ウェブサーバーをインストールします。

   Amazon Linux 2

   - Amazon Linux 2 で最新バージョンの NGINX をダウンロードする方法については、NGINX のウェブサイト を参照してください。

Amazon Linux 2 で入手可能な NGINX の最新バージョンは、システムバージョンの OpenSSL よりも新しいバージョンの OpenSSL を使用しています。NGINX をインストールしたら、 AWS CloudHSM OpenSSL Dynamic Engine ライブラリから、このバージョンの OpenSSL が想定する場所へのシンボリックリンクを作成する必要があります。

```
$ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
engines-1.1/cloudhsm.so
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Amazon Linux 2 の最新の Apache バージョンは、システムにインストールされているバージョンよりも新しい OpenSSL バージョンとリンクします。 AWS CloudHSM OpenSSL Dynamic Engine ライブラリから、新しいバージョンの OpenSSL が検索するパスへのシンボリックリンクを作成する必要があります。

```
$ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
engines-1.1/cloudhsm.so
```

Amazon Linux 2023

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 8

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 9

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Ubuntu 22.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 24.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 26.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

3. CloudHSM CLI を使用して Crypto User を作成します。HSM ユーザーの管理の詳細については、CloudHSM CLI を使用した HSM ユーザー管理について を参照してください。

> **Tip**
>
> CU のユーザー名とパスワードを書き留めます。後に、ウェブサーバーの HTTPS プライベートキーや証明書を生成またはインポートするときに必要になります。

以上のステップが完了したら、「ステップ 2: プライベートキーと SSL/TLS 証明書を生成する」に進みます。

注意事項

- セキュリティ強化 Linux (SELinux) および Web サーバーを使用するには、クライアント SDK 5 が HSM と通信するために使用するポート 2223 でアウトバウンド TCP 接続を許可する必要があります。
- クラスターを作成してアクティブ化し、EC2 インスタンスにクラスターへのアクセス権を付与するには、AWS CloudHSMの使用開始 の手順を実行します。「はじめに」では、1 つの HSM と Amazon EC2 クライアント インスタンスを含むアクティブなクラスターを作成するための段階的な手順が説明されています。このクライアントインスタンスをウェブサーバーとして使用することができます。
- クライアントキーの耐久性を無効化しないようにするには、クラスターに複数の HSM を追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。
- クライアントインスタンスに接続するには、SSH または PuTTY を使用することができます。詳細については、「Amazon EC2 ドキュメント」の「SSH を使用した Linux インスタンスへの接続」または「PuTTY を使用した Windows から Linux インスタンスへの接続」を参照してください。

ステップ 2: プライベートキーと SSL/TLS 証明書を生成する

HTTPS を有効にするには、ウェブサーバーアプリケーション (NGINX または Apache) にプライベートキーおよび対応する SSL/TLS 証明書が必要です。でウェブサーバーの SSL/TLS オフロードを使用するには AWS CloudHSM、プライベートキーを AWS CloudHSM クラスターの HSM に保存する必要があります。最初にプライベートキーを使用し、そのキーを使って証明書署名リクエスト (CSR) を作成します。次に HSM からフェイク PEM プライベートキーをエクスポートします。これは、HSM に保存されているプライベートキーへの参照を含む PEM 形式のプライベートキーファイルで、実際のプライベートキーではありません。ウェブサーバーは、SSL/TLS オフロード中にフェイク PEM プライベートキーファイルを使用して HSM のプライベートキーを識別します。

プライベートキーと証明書の生成

プライベートキーの生成

このセクションでは、CloudHSM CLI を使用してキーペアを生成する方法について説明します。HSM 内でキーペアを生成したら、それをフェイク PEM ファイルとしてエクスポートし、対応する証明書を生成できます。

CloudHSM CLI をインストールして設定する

1. CloudHSM CLI をインストールして設定します。
2. CloudHSM CLI を起動するには、次のコマンドを使用します。

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

3. 次のコマンドを実行して HSM にログインします。*<user name>* を Crypto User のユーザー名に置き換えます。

```
Command: login --username <user name> --role crypto-user
```

プライベートキーの生成

ユースケースに応じて、RSA または EC キーペアを生成できます。次のいずれかを行います。

- HSM で RSA プライベートキーを生成するには

  `key generate-asymmetric-pair rsa` コマンドを使用して RSA キーペアを生成します。この例では、modulus が 2,048、公開指数が 65,537、パブリックキーラベルが *tls_rsa_pub*、プライベートキーラベルが *tls_rsa_private* の RSA キーペアを生成します。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label tls_rsa_pub \
--private-label tls_rsa_private \
--private-attributes sign=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000280cc8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "tls_rsa_pub",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "public-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
```

```
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1
73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa
f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d5
133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e83469
ac3160f0ca9725d38318b7",
        "modulus-size-bits": 2048
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280cc7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "tls_rsa_private",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
```

```
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1
        "modulus-size-bits": 2048
      }
    }
  }
}
```

- HSM で EC プライベートキーを生成するには

  key generate-asymmetric-pair ec コマンドを使用して EC キーペアを生成します。この例では、prime256v1 曲線 (NID_X9_62_prime256v1 曲線に対応) を使用して EC キーペアを生成します。パブリックキーラベルは *tls_ec_pub*、プライベートキーラベルは *tls_ec_private* です。

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve prime256v1 \
    --public-label tls_ec_pub \
    --private-label tls_ec_private \
    --private-attributes sign=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
```

```
        "key-type": "ec",
        "label": "tls_ec_pub",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
        "curve": "secp224r1"
      }
    },
"private_key": {
      "key-reference": "0x000000000012000c",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "tls_ec_private",
```

```
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
        "curve": "secp224r1"
      }
    }
  }
}
```


フェイク PEM プライベートキーファイルをエクスポート

HSM にプライベートキーを作成したら、フェイク PEM プライベートキーファイルをエクスポートする必要があります。このファイルには実際のキーデータは含まれていませんが、OpenSSL Dynamic Engine が HSM 上のプライベートキーを識別できるようにします。その後、プライベートキーを使用して証明書署名リクエスト (CSR) を作成し、CSR に署名して証明書を作成できます。

`key generate-file` コマンドを使用して、プライベートキーをフェイク PEM 形式でエクスポートしてファイルに保存します。以下の値は独自の値に置き換えてください。

- *<private_key_label>* – 前のステップで生成したプライベートキーのラベル。
- *<web_server_fake_pem.key>* – フェイク PEM キーが書き込まれるファイルの名前。

```
aws-cloudhsm > key generate-file --encoding reference-pem --
path <web_server_fake_pem.key> --filter attr.label=<private_key_label>
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

CloudHSM CLI を終了する

CloudHSM CLI を停止するには、次のコマンドを実行します。

```
aws-cloudhsm > quit
```

これで、前のコマンドで *<web_server_fake_pem.key>* が指定したパスに新しいファイルがシステム上に作成されることになります。このファイルはフェイク PEM プライベートキーファイルです。

自己署名証明書を生成します

フェイク PEM プライベートキーを生成したら、このファイルを使用して証明書署名リクエスト (CSR) と証明書を生成できます。

本稼働環境では、通常、認証局 (CA) を使用して CSR から証明書を作成します。CA は、テスト環境では必要ありません。CA を使用する場合は、CA に CSR ファイルを送信し、HTTPS 用のウェブサーバーで提供される署名付き SSL/TLS 証明書を使用してください。

CA を使用する代わりに、 AWS CloudHSM OpenSSL Dynamic Engine を使用して自己署名証明書を作成できます。自己署名証明書はブラウザによって信頼されないため、本稼働環境では使用しないでください。これらは、テスト環境で使用することができます。

> **⚠ Warning**
>
> 自己署名証明書はテスト環境でのみ使用する必要があります。本稼働環境では、証明機関を使用して証明書を作成するなど、より安全な方法を使用してください。

OpenSSL Dynamic Engine をインストールして設定する

1. クライアントインスタンスに接続します。

2. the section called “インストール”

証明書を生成する

1. 以前のステップで生成したフェイク PEM ファイルのコピーを入手します。
2. CSR を作成する

   次のコマンドを実行して、 AWS CloudHSM OpenSSL Dynamic Engine を使用して証明書署名リクエスト (CSR) を作成します。*<web_server_fake_pem.key>* をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。*<web_server.csr>* を CSR が含まれるファイルの名前に置き換えます。

   `req` コマンドは対話的です。各フィールドに対応します。このフィールド情報は、SSL/TLS 証明書にコピーされます。

   ```
   $ openssl req -engine cloudhsm -new -key <web_server_fake_pem.key> -
   out <web_server.csr>
   ```

3. 自己署名証明書を作成する

   次のコマンドを実行して、 AWS CloudHSM OpenSSL Dynamic Engine を使用して HSM のプライベートキーで CSR に署名します。これにより、自己署名証明書が作成されます。コマンドの以下の値を独自の値に置き換えます。

   - *<web_server.csr>* - CSR を含むファイルの名前です。
   - *<web_server_fake_pem.key>* - フェイク PEM プライベートキーが含まれるファイルの名前です。
   - *<web_server.crt>* - ウェブサーバー証明書が含まれるファイルの名前です。

   ```
   $ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -
   signkey <web_server_fake_pem.key> -out <web_server.crt>
   ```

以上のステップが完了したら、「ステップ 3: ウェブサーバーを設定する」に進みます。

ステップ 3: ウェブサーバーを設定する

前のステップで作成した HTTPS 証明書とフェイク PEM プライベートキーを使用するようにウェブサーバーソフトウェアの設定を更新します。開始する前に、既存の証明書とキーを必ずバックアップしてください。これで、 AWS CloudHSMを使用して、Linux ウェブサーバーソフトウェアに SSL/TLS オフロードを設定できます。

次のいずれかのセクションの手順を完了します。

トピック

- NGINX ウェブサーバーを設定する
- Apache ウェブサーバーの設定をします。

NGINX ウェブサーバーを設定する

このセクションは、サポートされているプラットフォームで NGINX を設定するために使用します。

NGINX のウェブサーバー設定を更新するには

1. クライアントインスタンスに接続します。
2. 次のコマンドを実行して、ウェブサーバー証明書とフェイク PEM プライベートキーに必要なディレクトリを作成します。

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. 次のコマンドを実行して、ウェブサーバーの証明書を所定場所にコピーします。*<web_server.crt>* を、ウェブサーバー証明書の名前に置き換えます。

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. 次のコマンドを実行して、フェイク PEM プライベートキーを所定場所にコピーします。*<web_server_fake_pem.key>* をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/nginx/private/server.key
```

5. 次のコマンドを実行してファイルの所有権を変更し、nginx という名前のユーザーがそれらのファイルを読み取れるようにします。

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. 次のコマンドを実行して、/etc/nginx/nginx.conf ファイルをバックアップします。

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. NGINX の設定を更新します。

> **Note**
>
> 各クラスターは、すべての NGINX ウェブサーバーで最大 1,000 の NGINX ワーカープロセスをサポートできます。

Amazon Linux 2

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには Linux の root 権限が必要です。ファイルの先頭に、次の行を追加します。

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
```

```
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Amazon Linux 2023

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには Linux の root 権限が必要です。ファイルの先頭に、次の行を追加します。

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
```

```
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Red Hat 8

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには Linux の root 権限が必要です。ファイルの先頭に、次の行を追加します。

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
```

```
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Red Hat 9

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには Linux の root 権限が必要です。ファイルの先頭に、次の行を追加します。

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Ubuntu 22.04 LTS

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには Linux の root 権限が必要です。ファイルの先頭に、次の行を追加します。

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
```

```
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Ubuntu 24.04 LTS

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには Linux の root 権限が必要です。ファイルの先頭に、次の行を追加します。

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;
```

```
    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Ubuntu 26.04 LTS

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには Linux の root 権限が必要です。ファイルの先頭に、次の行を追加します。

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
```

```
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

ファイルを保存します。

8. systemd 設定ファイルをバックアップしてから、EnvironmentFile パスを設定します。

   Amazon Linux 2

   1. nginx.service ファイルをバックアップします。

      ```
      $ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
      nginx.service.backup
      ```

   2. /lib/systemd/system/nginx.service ファイルをテキストエディタで開き、[Service] セクションに次のパスを追加します。

      ```
      EnvironmentFile=/etc/sysconfig/nginx
      ```

Amazon Linux 2023

1. nginx.service ファイルをバックアップします。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. テキストエディタで /lib/systemd/system/nginx.service ファイルを開き、〔サービス] セクションで次のパスを追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 8

1. nginx.service ファイルをバックアップします。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. /lib/systemd/system/nginx.service ファイルをテキストエディタで開き、[Service] セクションに次のパスを追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 9

1. nginx.service ファイルをバックアップします。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. /lib/systemd/system/nginx.service ファイルをテキストエディタで開き、[Service] セクションに次のパスを追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

1. nginx.service ファイルをバックアップします。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. /lib/systemd/system/nginx.service ファイルをテキストエディタで開き、[Service] セクションに次のパスを追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 24.04 LTS

1. nginx.service ファイルをバックアップします。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. /lib/systemd/system/nginx.service ファイルをテキストエディタで開き、[Service] セクションに次のパスを追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 26.04 LTS

1. nginx.service ファイルをバックアップします。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. /lib/systemd/system/nginx.service ファイルをテキストエディタで開き、[Service] セクションに次のパスを追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

9. /etc/sysconfig/nginx ファイルの存在を確認してから、次のいずれかを実行します。

- ファイルが存在する場合は、次のコマンドを実行してファイルをバックアップします。

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- ファイルが存在しない場合は、テキストエディタを開き、/etc/sysconfig/ フォルダ内に nginx という名前のファイルを作成します。

10. NGINX 環境を設定します。

> **Note**
>
> クライアント SDK 5 では CU の認証情報を保存するための CLOUDHSM_PIN 環境変数が導入されています。

Amazon Linux 2

テキストエディタで /etc/sysconfig/nginx ファイルを開きます。これには Linux の root 権限が必要です。Cryptography User (CU) 認証情報を追加:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

ファイルを保存します。

Amazon Linux 2023

テキストエディタで /etc/sysconfig/nginx ファイルを開きます。これには Linux の root 権限が必要です。Cryptography User (CU) 認証情報を追加:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

ファイルを保存します。

Red Hat 8

テキストエディタで /etc/sysconfig/nginx ファイルを開きます。これには Linux の root 権限が必要です。Cryptography User (CU) 認証情報を追加:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

ファイルを保存します。

Red Hat 9

テキストエディタで `/etc/sysconfig/nginx` ファイルを開きます。これには Linux の root 権限が必要です。Cryptography User (CU) 認証情報を追加:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

ファイルを保存します。

Ubuntu 22.04 LTS

テキストエディタで `/etc/sysconfig/nginx` ファイルを開きます。これには Linux の root 権限が必要です。Cryptography User (CU) 認証情報を追加:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

ファイルを保存します。

Ubuntu 24.04 LTS

テキストエディタで `/etc/sysconfig/nginx` ファイルを開きます。これには Linux の root 権限が必要です。Cryptography User (CU) 認証情報を追加:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

ファイルを保存します。

Ubuntu 26.04 LTS

テキストエディタで /etc/sysconfig/nginx ファイルを開きます。これには Linux の root 権限が必要です。Cryptography User (CU) 認証情報を追加:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

ファイルを保存します。

11. NGINX ウェブサーバーを起動します。

Amazon Linux 2

実行中の NGINX プロセスをすべて停止する

```
$ sudo systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ sudo systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ sudo systemctl start nginx
```

Amazon Linux 2023

すべての NGINX プロセスを停止する

```
$ sudo systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ sudo systemctl daemon-reload
```

NGINX を開始する

```
$ sudo systemctl start nginx
```

Red Hat 8

実行中の NGINX プロセスをすべて停止する

```
$ sudo systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ sudo systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ sudo systemctl start nginx
```

Red Hat 9

実行中の NGINX プロセスをすべて停止する

```
$ sudo systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ sudo systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

実行中の NGINX プロセスをすべて停止する

```
$ sudo systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ sudo systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ sudo systemctl start nginx
```

Ubuntu 24.04 LTS

実行中の NGINX プロセスをすべて停止する

```
$ sudo systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ sudo systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ sudo systemctl start nginx
```

Ubuntu 26.04 LTS

実行中の NGINX プロセスをすべて停止する

```
$ sudo systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ sudo systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ sudo systemctl start nginx
```

12. （オプション）スタートアップ時に NGINX を起動するようにプラットフォームを設定します。

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

Amazon Linux 2023

```
$ sudo systemctl enable nginx
```

Red Hat 8

```
$ sudo systemctl enable nginx
```

Red Hat 9

```
$ sudo systemctl enable nginx
```

Ubuntu 22.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 24.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 26.04 LTS

```
$ sudo systemctl enable nginx
```

ウェブサーバー設定を更新したら、「ステップ 4: HTTPS トラフィックを有効にして証明書を検証する」に移動します。

Apache ウェブサーバーの設定をします。

このセクションでは、サポートされているプラットフォームで Apache を設定します。

Apache のウェブサーバー設定を更新するには

1. Amazon EC2 クライアントインスタンスに接続します。

2. プラットフォーム用の証明書とプライベートキーのデフォルトの場所を定義します。

Amazon Linux 2

/etc/httpd/conf.d/ssl.conf ファイルに、次の値が存在することを確認します。

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Amazon Linux 2023

/etc/httpd/conf.d/ssl.conf ファイルで、次の値が存在することを確認します。

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

/etc/httpd/conf.d/ssl.conf ファイルに、次の値が存在することを確認します。

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 9

/etc/httpd/conf.d/ssl.conf ファイルに、次の値が存在することを確認します。

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Ubuntu 22.04 LTS

/etc/apache2/sites-available/default-ssl.conf ファイルに、次の値が存在することを確認します。

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 24.04 LTS

/etc/apache2/sites-available/default-ssl.conf ファイルに、次の値が存在することを確認します。

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 26.04 LTS

/etc/apache2/sites-available/default-ssl.conf ファイルに、次の値が存在することを確認します。

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

3. ウェブサーバーの証明書を、プラットフォームで必要な場所にコピーします。

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

<web_server.crt> を、ウェブサーバー証明書の名前に置き換えます。

Amazon Linux 2023

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

<web_server.crt> を、ウェブサーバー証明書の名前に置き換えます。

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

<web_server.crt> を、ウェブサーバー証明書の名前に置き換えます。

Red Hat 9

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

<web_server.crt> を、ウェブサーバー証明書の名前に置き換えます。

Ubuntu 22.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

<web_server.crt> を、ウェブサーバー証明書の名前に置き換えます。

Ubuntu 24.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

<web_server.crt> を、ウェブサーバー証明書の名前に置き換えます。

Ubuntu 26.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

<web_server.crt> を、ウェブサーバー証明書の名前に置き換えます。

4. 偽の PEM プライベートキーをプラットフォームの所定場所にコピーします。

Amazon Linux 2

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

<web_server_fake_pem.key> をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

Amazon Linux 2023

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

<web_server_fake_pem.key> をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

Red Hat 8

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

<web_server_fake_pem.key> をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

Red Hat 9

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

<web_server_fake_pem.key> をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

Ubuntu 22.04 LTS

```
$ sudo cp <web_server_fake_pem.key> /etc/ssl/private/localhost.key
```

<web_server_fake_pem.key> をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

Ubuntu 24.04 LTS

```
$ sudo cp <web_server_fake_pem.key> /etc/ssl/private/localhost.key
```

<web_server_fake_pem.key> をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

Ubuntu 26.04 LTS

```
$ sudo cp <web_server_fake_pem.key> /etc/ssl/private/localhost.key
```

<web_server_fake_pem.key> をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

5. プラットフォームで必要な場合は、これらのファイルの所有権を変更します。

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache という名前のユーザーに読み取り権限を与えます。

Amazon Linux 2023

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache という名前のユーザーに読み取り権限を与えます。

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache という名前のユーザーに読み取り権限を与えます。

Red Hat 9

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache という名前のユーザーに読み取り権限を与えます。

Ubuntu 22.04 LTS

対処は必要ありません。

Ubuntu 24.04 LTS

対処は必要ありません。

Ubuntu 26.04 LTS

対処は必要ありません。

6. プラットフォームに合わせて、Apache のディレクティブを設定します。

Amazon Linux 2

このプラットフォームの SSL ファイルを探します。

```
/etc/httpd/conf.d/ssl.conf
```

このファイルには、サーバーの実行方法を定義する Apache ディレクティブが含まれています。ディレクティブは左側に表示され、その後に値が続きます。テキストエディタを使用して、このファイルを編集します。これには Linux の root 権限が必要です。

これらの値を使用して、次のディレクティブを更新または入力します。

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

ファイルを保存します。

Amazon Linux 2023

このプラットフォームの SSL ファイルを探します。

```
/etc/httpd/conf.d/ssl.conf
```

このファイルには、サーバーの実行方法を定義する Apache ディレクティブが含まれています。ディレクティブは左側に表示され、その後に値が続きます。テキストエディタを使用して、このファイルを編集します。これには Linux の root 権限が必要です。

これらの値を使用して、次のディレクティブを更新または入力します。

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

ファイルを保存します。

Red Hat 8

このプラットフォームの SSL ファイルを探します。

```
/etc/httpd/conf.d/ssl.conf
```

このファイルには、サーバーの実行方法を定義する Apache ディレクティブが含まれています。ディレクティブは左側に表示され、その後に値が続きます。テキストエディタを使用して、このファイルを編集します。これには Linux の root 権限が必要です。

これらの値を使用して、次のディレクティブを更新または入力します。

```
SSLCryptoDevice cloudhsm
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProxyCipherSuite HIGH:!aNULL
```

ファイルを保存します。

Red Hat 9

このプラットフォームの SSL ファイルを探します。

```
/etc/httpd/conf.d/ssl.conf
```

このファイルには、サーバーの実行方法を定義する Apache ディレクティブが含まれています。ディレクティブは左側に表示され、その後に値が続きます。テキストエディタを使用して、このファイルを編集します。これには Linux の root 権限が必要です。

これらの値を使用して、次のディレクティブを更新または入力します。

```
SSLCryptoDevice cloudhsm
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

```
SSLProxyCipherSuite HIGH:!aNULL
```

ファイルを保存します。

Ubuntu 22.04 LTS

このプラットフォームの SSL ファイルを探します。

```
/etc/apache2/mods-available/ssl.conf
```

このファイルには、サーバーの実行方法を定義する Apache ディレクティブが含まれています。ディレクティブは左側に表示され、その後に値が続きます。テキストエディタを使用して、このファイルを編集します。これには Linux の root 権限が必要です。

これらの値を使用して、次のディレクティブを更新または入力します。

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

ファイルを保存します。

SSL モジュールとデフォルトの SSL サイト設定を有効にします。

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 24.04 LTS

このプラットフォームの SSL ファイルを探します。

```
/etc/apache2/mods-available/ssl.conf
```

このファイルには、サーバーの実行方法を定義する Apache ディレクティブが含まれています。ディレクティブは左側に表示され、その後に値が続きます。テキストエディタを使用して、このファイルを編集します。これには Linux の root 権限が必要です。

これらの値を使用して、次のディレクティブを更新または入力します。

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

ファイルを保存します。

SSL モジュールとデフォルトの SSL サイト設定を有効にします。

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 26.04 LTS

このプラットフォームの SSL ファイルを探します。

```
/etc/apache2/mods-available/ssl.conf
```

このファイルには、サーバーの実行方法を定義する Apache ディレクティブが含まれています。ディレクティブは左側に表示され、その後に値が続きます。テキストエディタを使用して、このファイルを編集します。これには Linux の root 権限が必要です。

これらの値を使用して、次のディレクティブを更新または入力します。

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
```

```
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

ファイルを保存します。

SSL モジュールとデフォルトの SSL サイト設定を有効にします。

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

7. プラットフォーム用の環境値ファイルを設定します。

Amazon Linux 2

httpd サービスファイルを開きます。

```
/lib/systemd/system/httpd.service
```

[Service] セクションの下に、以下を追加します。

```
EnvironmentFile=/etc/sysconfig/httpd
```

Amazon Linux 2023

httpd サービスファイルを開きます。

```
/lib/systemd/system/httpd.service
```

[Service] セクションの下に、以下を追加します。

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

httpd サービスファイルを開きます。

```
/lib/systemd/system/httpd.service
```

[Service] セクションの下に、以下を追加します。

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 9

httpd サービスファイルを開きます。

```
/lib/systemd/system/httpd.service
```

[Service] セクションの下に、以下を追加します。

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 22.04 LTS

対処は必要ありません。/etc/sysconfig/httpd に環境値が入ります

Ubuntu 24.04 LTS

対処は必要ありません。/etc/sysconfig/httpd に環境値が入ります

Ubuntu 26.04 LTS

対処は必要ありません。/etc/sysconfig/httpd に環境値が入ります

8. プラットフォーム用の環境変数を格納するファイルで、暗号化ユーザー (CU) の認証情報を含む環境変数を設定します。

Amazon Linux 2

テキストエディタを使用して、/etc/sysconfig/httpd を編集します。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

Amazon Linux 2023

テキストエディタを使用して、/etc/sysconfig/httpd を編集します。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

Red Hat 8

テキストエディタを使用して、/etc/sysconfig/httpd を編集します。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

> **Note**
>
> クライアント SDK 5 では CU の認証情報を保存するための CLOUDHSM_PIN 環境変数が導入されています。

Red Hat 9

テキストエディタを使用して、/etc/sysconfig/httpd を編集します。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

> **Note**
>
> クライアント SDK 5 では CU の認証情報を保存するための CLOUDHSM_PIN 環境変数が導入されています。

Ubuntu 22.04 LTS

テキストエディタを使用して、/etc/apache2/envvars を編集します。

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

> **Note**
>
> クライアント SDK 5 では CU の認証情報を保存するための CLOUDHSM_PIN 環境変数が導入されています。クライアント SDK 3 では、CU の認証情報を n3fips_password 環境変数に保存していました。クライアント SDK 5 は両方の環境変数をサポートしますが、CLOUDHSM_PIN を使用することを推奨します。

Ubuntu 24.04 LTS

テキストエディタを使用して、/etc/apache2/envvars を編集します。

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

> **Note**
>
> クライアント SDK 5 では CU の認証情報を保存するための CLOUDHSM_PIN 環境変数が導入されています。クライアント SDK 3 では、CU の認証情報を n3fips_password 環境変数に保存していました。クライアント SDK 5 は両方の環境変数をサポートしますが、CLOUDHSM_PIN を使用することを推奨します。

Ubuntu 26.04 LTS

テキストエディタを使用して、/etc/apache2/envvars を編集します。

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。

> **Note**
>
> クライアント SDK 5 では CU の認証情報を保存するための CLOUDHSM_PIN 環境変数が導入されています。クライアント SDK 3 では、CU の認証情報を

`n3fips_password` 環境変数に保存していました。クライアント SDK 5 は両方の環境変数をサポートしますが、`CLOUDHSM_PIN` を使用することを推奨します。

9. Apache ウェブサーバーを起動します。

   Amazon Linux 2

   ```
   $ sudo systemctl daemon-reload
   $ sudo service httpd start
   ```

   Amazon Linux 2023

   ```
   $ sudo systemctl daemon-reload
   $ sudo service httpd start
   ```

   Red Hat 8

   ```
   $ sudo systemctl daemon-reload
   $ sudo service httpd start
   ```

   Red Hat 9

   ```
   $ sudo systemctl daemon-reload
   $ sudo service httpd start
   ```

   Ubuntu 22.04 LTS

   ```
   $ sudo service apache2 start
   ```

   Ubuntu 24.04 LTS

   ```
   $ sudo service apache2 start
   ```

   Ubuntu 26.04 LTS

   ```
   $ sudo service apache2 start
   ```

10. (オプション) スタートアップに Apache を起動するようにプラットフォームを設定します。

Amazon Linux 2

```
$ sudo chkconfig httpd on
```

Amazon Linux 2023

```
$ sudo chkconfig httpd on
```

Red Hat 8

```
$ systemctl enable httpd
```

Red Hat 9

```
$ systemctl enable httpd
```

Ubuntu 22.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 24.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 26.04 LTS

```
$ sudo systemctl enable apache2
```

ウェブサーバー設定を更新したら、「ステップ 4: HTTPS トラフィックを有効にして証明書を検証する」に移動します。

ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

で SSL/TLS オフロード用にウェブサーバーを設定したら AWS CloudHSM、インバウンド HTTPS トラフィックを許可するセキュリティグループにウェブサーバーインスタンスを追加します。これにより、ウェブブラウザなどのクライアントがウェブサーバーと HTTPS 接続を確立できるようになり

ます。次に、ウェブサーバーへの HTTPS 接続を行い、SSL/TLS オフロード用に設定した証明書を使用していることを確認します AWS CloudHSM。

トピック

- インバウンド HTTPS 接続の有効化
- 設定した証明書が HTTPS で使用されていることを検証する

インバウンド HTTPS 接続の有効化

クライアント (ウェブブラウザなど) からウェブサーバーに接続するには、インバウンド HTTPS 接続を許可するセキュリティグループを作成します。具体的には、ポート 443 でインバウンドの TCP 接続を許可する必要があります。このセキュリティグループをウェブサーバーに割り当てます。

HTTPS のセキュリティグループを作成してウェブサーバーに割り当てるには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
2. ナビゲーションペインで、[セキュリティグループ] を選択します。
3. [Create Security Group] を選択します。
4. [Create Security Group] で、以下の操作を行います。
    a. [Security group name] に、作成するセキュリティグループの名前を入力します
    b. (オプション) 作成するセキュリティグループの説明を入力します。
    c. [VPC] で、ウェブサーバーのAmazon EC2インスタンスが含まれている VPC を選択します。
    d. [Add rule (ルールの追加)] を選択します。
    e. [タイプ] で、ドロップダウンウィンドウから [HTTPS] を選択します。
    f. [ソース] には、ソースの場所を入力します。
    g. [セキュリティグループの作成] を選択してください。
5. ナビゲーションペインで、[インスタンス] を選択してください。
6. ウェブサーバーインスタンスの横にあるチェックボックスを選択します。
7. ページの上部で [アクション] ドロップダウンメニューを選択します。[セキュリティ] を選択し、[セキュリティグループの変更] を選択します。
8. [関連付けられたセキュリティグループ] で、検索ボックスを選択して HTTPS 用に作成したセキュリティグループを選択します。次に、[セキュリティグループの追加] を選択します。
9. [保存] を選択します。

設定した証明書が HTTPS で使用されていることを検証する

ウェブサーバーをセキュリティグループに追加した後、SSL/TLS オフロードが自己署名証明書を使用していることを確認できます。この検証には、ウェブブラウザ、または OpenSSL s_client などのツールを使用できます。

ウェブブラウザで SSL/TLS オフロードを確認するには

1. ウェブブラウザを使用し、サーバーの公開 DNS 名または IP アドレスを使用してウェブサーバーに接続します。アドレスバーの URL が https:// で始まっていることを確認します。例えば、**`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. ウェブブラウザを使用して、ウェブサーバー証明書を表示します。詳細については次を参照してください:

   - Mozilla Firefox の場合は、Mozilla サポートウェブサイトの「証明書を見る」を参照してください。
   - Google Chrome の場合は、ウェブ開発者向け Google ツールのウェブサイトで「セキュリティの問題を理解する」を参照してください。

   他のウェブブラウザでも、同様の機能を使用してウェブサーバー証明書を表示できる場合があります。

3. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

OpenSSL s_client で SSL/TLS オフロードを確認するには

1. HTTPS を使用してウェブサーバーに接続するには、次の OpenSSL コマンドを実行します。*<#####>* は、ウェブサーバーの公開 DNS 名または IP アドレスに置き換えます。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

これで、ウェブサイトが HTTPS で保護されるようになりました。ウェブサーバーのプライベートキーは、 AWS CloudHSM クラスターの HSM に保存されます。

ロードバランサーを追加するには、「Elastic Load Balancing for でロードバランサーを追加する AWS CloudHSM(オプション)」を参照してください。

AWS CloudHSM OpenSSL プロバイダーで NGINX または HAProxy を使用した Linux での OpenSSL/TLS オフロード

このトピックでは、OpenSSL プロバイダーで NGINX または HAProxy を使用して Linux ウェブサーバー AWS CloudHSM 上の で SSL/TLS サーバー ID OpenSSL オフロードを設定するstep-by-stepについて説明します。

トピック

- 概要
- ステップ 1: 前提条件の設定
- ステップ 2: プライベートキーを生成またはインポートし、証明書を取得する
- ステップ 3: ウェブサーバーを設定する
- ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

概要

Linux では、NGINX および HAProxy ウェブサーバーソフトウェアは OpenSSL と統合して HTTPS をサポートします。AWS CloudHSM OpenSSL プロバイダーは、ウェブサーバーソフトウェアがクラスター内の HSMs を暗号化オフロードとキーストレージに使用できるインターフェイスを提供します。OpenSSL プロバイダーは、ウェブサーバーを AWS CloudHSM クラスターに接続するブリッジです。

このチュートリアルを完了するには、 AWS CloudHSM OpenSSL プロバイダーを使用するように NGINX または HAProxy を設定します。このチュートリアルでは、以下の方法を示します。

- Amazon EC2 インスタンスに、ウェブサーバーソフトウェアをインストールします。
- AWS CloudHSM クラスターに保存されている秘密キーで、HTTPS をサポートするようにウェブサーバソフトウェアを設定します。
- (オプション) Amazon EC2 を使用して 2 台目のウェブサーバーインスタンスを作成し、Elastic Load Balancing を使用してロードバランサーを作成します。ロードバランサーを使用すると、複数のサーバーに負荷を分散することでパフォーマンスを向上させることができます。また、1 つ以上のサーバーに障害が発生した場合、冗長性と高可用性を提供します。

始める準備ができたら、「ステップ 1: 前提条件の設定」を参照してください。

ステップ 1: 前提条件の設定

プラットフォームごとに、異なる前提条件が必要です。以下の前提条件セクションのうち、お使いのプラットフォームに合ったものをご利用ください。

AWS CloudHSM OpenSSL プロバイダーの前提条件

クライアント SDK 5 の AWS CloudHSM OpenSSL プロバイダーを使用してウェブサーバー SSL/TLS サーバー ID オフロードを設定するには、以下が必要です。

- 少なくとも 2 つのハードウェアセキュリティモジュール (HSM) を持つアクティブな AWS CloudHSM クラスター

> **Note**
>
> HSM クラスターは1つでも使用できますが、まずクライアントキーの耐久性を無効にする必要があります。詳細については、クライアントキーの耐久性設定の管理 そして クライアント SDK 5 設定ツール を参照してください。

- Amazon EC2 インスタンスが Linux オペレーティングシステムを実行します。インスタンスに次のソフトウェアがインストールされていることを確認します。
  - ウェブサーバー (NGINX または HAProxy)
  - クライアント SDK 5 の AWS CloudHSM OpenSSL プロバイダー
- HSM でこのウェブサーバーのプライベートキーを所有および管理する Crypto User (CU)。

Linux ウェブサーバーインスタンスをセットアップし、HSM で CU を作成するには

> **Note**
>
> この手順のコマンドの多くは、昇格された権限を必要とします。システム設定によっては、ルートユーザーとして sudoまたは を使用してコマンドを実行する必要がある場合があります。

1. クライアント SDK 5 の AWS CloudHSM OpenSSL プロバイダーをインストールして設定します。OpenSSL プロバイダーのインストールの詳細については、「AWS CloudHSM OpenSSL Provider for Client SDK 5」を参照してください。
2. クラスターにアクセスできる EC2 Linux インスタンスで、NGINX または HAProxy ウェブサーバーをインストールします。

   Amazon Linux 2023

   - NGINX

     ```
     $ yum install nginx
     ```

   - HAProxy

     ```
     $ yum install haproxy
     ```

   RHEL 9 (9.2+)

   - NGINX

     ```
     $ yum install nginx
     ```

   - HAProxy

     ```
     $ yum install haproxy
     ```

   RHEL 10 (10.0+)

   - NGINX

```
$ yum install nginx
```

- HAProxy

```
$ yum install haproxy
```

Ubuntu 26.04 LTS

- NGINX

```
$ apt install nginx
```

- HAProxy

```
$ apt install haproxy
```

3. CloudHSM CLI を使用して Crypto User を作成します。HSM ユーザーの管理の詳細については、CloudHSM CLI を使用した HSM ユーザー管理について を参照してください。

> **Tip**
>
> CU のユーザー名とパスワードを書き留めます。後に、ウェブサーバーの HTTPS プライベートキーや証明書を生成またはインポートするときに必要になります。

以上のステップが完了したら、「ステップ 2: プライベートキーを生成またはインポートし、証明書を取得する」に進みます。

注意事項

- セキュリティ強化 Linux (SELinux) および Web サーバーを使用するには、クライアント SDK 5 が HSM と通信するために使用するポート 2223 でアウトバウンド TCP 接続を許可する必要があります。
- クラスターを作成してアクティブ化し、EC2 インスタンスにクラスターへのアクセス権を付与するには、AWS CloudHSMの使用開始 の手順を実行します。「はじめに」では、1 つの HSM と Amazon EC2 クライアント インスタンスを含むアクティブなクラスターを作成するための段階的な手順が説明されています。このクライアントインスタンスをウェブサーバーとして使用することができます。

- クライアントキーの耐久性を無効化しないようにするには、クラスターに複数の HSM を追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。
- クライアントインスタンスに接続するには、SSH または PuTTY を使用することができます。詳細については、「Amazon EC2 ドキュメント」の「SSH を使用した Linux インスタンスへの接続」または「PuTTY を使用した Windows から Linux インスタンスへの接続」を参照してください。

ステップ 2: プライベートキーを生成またはインポートし、証明書を取得する

HTTPS を有効にするには、ウェブサーバーアプリケーション (NGINX または HAProxy) にプライベートキーと対応する SSL/TLS 証明書が必要です。でウェブサーバー SSL/TLS サーバー ID オフロードを使用するには AWS CloudHSM、プライベートキーを AWS CloudHSM クラスターの HSM に保存する必要があります。最初にプライベートキーを使用し、そのキーを使って証明書署名リクエスト (CSR) を作成します。次に HSM からフェイク PEM プライベートキーをエクスポートします。これは、HSM に保存されているプライベートキーへの参照を含む PEM 形式のプライベートキーファイルで、実際のプライベートキーではありません。ウェブサーバーは、フェイク PEM プライベートキーファイルを使用して、SSL/TLS サーバー ID オフロード中に HSM のプライベートキーを識別します。

プライベートキーの生成

このセクションでは、CloudHSM CLI を使用してキーペアを生成する方法について説明します。HSM 内でキーペアを生成したら、それをフェイク PEM ファイルとしてエクスポートし、対応する証明書を生成できます。

CloudHSM CLI をインストールして設定する

1. CloudHSM CLI をインストールして設定します。
2. CloudHSM CLI を起動するには、次のコマンドを使用します。

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

3. 次のコマンドを実行して HSM にログインします。*<user name>* を Crypto User のユーザー名に置き換えます。

```
aws-cloudhsm>login --username <user name> --role crypto-user
```

プライベートキーの生成

ユースケースに応じて、RSA または EC キーペアを生成できます。TLS 1.3 で使用する Ed25519 または ML-DSA キーペアを生成することもできます。次のいずれかを行います:

- HSM で RSA プライベートキーを生成するには

  key generate-asymmetric-pair rsa コマンドを使用して RSA キーペアを生成します。この例では、modulus が 2,048、公開指数が 65,537、パブリックキーラベルが *tls_rsa_pub*、プライベートキーラベルが *tls_rsa_private* の RSA キーペアを生成します。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label tls_rsa_pub \
--private-label tls_rsa_private \
--private-attributes sign=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000280cc8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "tls_rsa_pub",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "public-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
```

```
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
"0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1
        "modulus-size-bits": 2048
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280cc7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "tls_rsa_private",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1
        "modulus-size-bits": 2048
      }
    }
  }
}
```

- HSM で EC プライベートキーを生成するには

  key generate-asymmetric-pair ec コマンドを使用して EC キーペアを生成します。この例では、prime256v1 曲線 (NID_X9_62_prime256v1 曲線に対応) を使用して EC キーペアを生成します。パブリックキーラベルは *tls_ec_pub*、プライベートキーラベルは *tls_ec_private* です。

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve prime256v1 \
    --public-label tls_ec_pub \
    --private-label tls_ec_private \
    --private-attributes sign=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
```

```
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "tls_ec_pub",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
        "curve": "secp224r1"
      }
    },
"private_key": {
      "key-reference": "0x000000000012000c",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
```

```
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "tls_ec_private",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
        "curve": "secp224r1"
      }
    }
  }
}
```

- HSM で Ed25519 プライベートキーを生成するには

> **Ed25519 の可用性**
>
> Ed25519 は、FIPS 以外のクラスターでのみ使用できます。Ed25519 は TLS 1.3 のみをサポートしています。

key generate-asymmetric-pair ec コマンドを使用して Ed25519 キーペアを生成します。この例では、パブリックキーラベルが *tls_ed*Ed25519*25519* キーペアを生成します。

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve ed25519 \
    --public-label tls_ed25519_pub \
    --private-label tls_ed25519_private \
    --private-attributes sign=true \
    --public-attributes verify=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000383931",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "tls_ed25519_pub",
        "id": "0x",
        "check-value": "0x4cb1b1",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 32,
        "ec-point":
"0x302a300506032b6570032100c31f8a3695f571dda1658bc98e0591ab088c1a657b3199dd50c3956046ebbe75"
        "curve": "ed25519"
      }
    },
    "private_key": {
      "key-reference": "0x0000000000382e60",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "tls_ed25519_private",
        "id": "0x",
        "check-value": "0x4cb1b1",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
```

```
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 48,
        "ec-point":
 "0x302a300506032b6570032100c31f8a3695f571dda1658bc98e0591ab088c1a657b3199dd50c3956046ebbe75"
        "curve": "ed25519"
      }
    }
  }
}
```

- HSM で ML-DSA プライベートキーを生成するには

> **ML-DSA の可用性と要件**
>
> ML-DSA では、CSR の作成、証明書の作成、TLS オフロードに OpenSSL 3.5 以降が必要です。ML-DSA は TLS 1.3 のみをサポートしています。
> Amazon Linux 2023 および RHEL プラットフォームでの ML-DSA TLS オフロードの場合、ポスト量子 (PQ) 暗号化サブポリシーを有効にする必要がある場合があります。詳細については、「the section called “問題: Amazon Linux 2023 と RHEL で「共有署名アルゴリズムなし」で ML-DSA TLS ハンドシェイクが失敗する”」を参照してください。

`key generate-asymmetric-pair ml-dsa` コマンドを使用して ML-DSA キーペアを生成します。この例では、パブリックキーラベルが *`tls_mldsa`*ML-DSA-44*`-44`* キーペアを生成します。ML-DSA-65 または を使用して、より高いセキュリティレベルML-DSA-87にすることもできます。

```
aws-cloudhsm > key generate-asymmetric-pair ml-dsa \
    --mldsa-algorithm ML-DSA-44 \
    --public-label tls_mldsa44_pub \
```

```
    --private-label tls_mldsa44_private \
    --private-attributes sign=true \
    --public-attributes verify=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000280cd2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ml-dsa",
        "label": "tls_mldsa44_pub",
        "id": "",
        "check-value": "0xd4e5f6",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1312,
        "mldsa-algorithm": "ML-DSA-44"
```

```
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280cd3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ml-dsa",
        "label": "tls_mldsa44_private",
        "id": "",
        "check-value": "0xd4e5f6",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 2560,
        "mldsa-algorithm": "ML-DSA-44"
      }
    }
  }
```

```
}
```

フェイク PEM プライベートキーファイルをエクスポート

HSM にプライベートキーを作成したら、フェイク PEM プライベートキーファイルをエクスポートする必要があります。このファイルには実際のキーデータは含まれませんが、OpenSSL プロバイダーは HSM のプライベートキーを識別できます。その後、プライベートキーを使用して証明書署名リクエスト (CSR) を作成し、CSR に署名して証明書を作成できます。

key generate-file コマンドを使用して、プライベートキーをフェイク PEM 形式でエクスポートしてファイルに保存します。以下の値は独自の値に置き換えてください。

- *<private_key_label>* – 前のステップで生成したプライベートキーのラベル。
- *<web_server_fake_pem.key>* – フェイク PEM キーが書き込まれるファイルの名前。

RSA キーと EC キーの場合

reference-pem エンコードを使用します。

```
aws-cloudhsm > key generate-file --encoding reference-pem --
path <web_server_fake_pem.key> --filter attr.label=<private_key_label>
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

Ed25519 キーと ML-DSA キーの場合

cloudhsm-reference-pem エンコードを使用します。

```
aws-cloudhsm > key generate-file --encoding cloudhsm-reference-pem --
path <web_server_fake_pem.key> --filter attr.label=<private_key_label>
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

CloudHSM CLI を終了する

CloudHSM CLI を停止するには、次のコマンドを実行します。

```
aws-cloudhsm > quit
```

これで、前のコマンドで *<web_server_fake_pem.key>* が指定したパスに新しいファイルがシステム上に作成されることになります。このファイルはフェイク PEM プライベートキーファイルです。

自己署名証明書を生成します

フェイク PEM プライベートキーを生成したら、このファイルを使用して証明書署名リクエスト (CSR) と証明書を生成できます。

本稼働環境では、通常、認証局 (CA) を使用して CSR から証明書を作成します。CA は、テスト環境では必要ありません。CA を使用する場合は、CA に CSR ファイルを送信し、HTTPS 用のウェブサーバーで提供される署名付き SSL/TLS 証明書を使用してください。

CA を使用する代わりに、 AWS CloudHSM OpenSSL プロバイダーを使用して自己署名証明書を作成できます。自己署名証明書はブラウザによって信頼されないため、本稼働環境では使用しないでください。これらは、テスト環境で使用することができます。

> **⚠ Warning**
>
> 自己署名証明書はテスト環境でのみ使用する必要があります。本稼働環境では、証明機関を使用して証明書を作成するなど、より安全な方法を使用してください。

OpenSSL プロバイダーのインストールと設定

1. クライアントインスタンスに接続します。
2. the section called “インストール”

証明書を生成する

1. 以前のステップで生成したフェイク PEM ファイルのコピーを入手します。
2. CSR を作成する

次のコマンドを実行して、 AWS CloudHSM OpenSSL プロバイダーを使用して証明書署名リクエスト (CSR) を作成します。*<web_server_fake_pem.key>* をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。*<web_server.csr>* を CSR が含まれるファイルの名前に置き換えます。

`req` コマンドは対話的です。各フィールドに対応します。このフィールド情報は、SSL/TLS 証明書にコピーされます。

> **ML-DSA OpenSSL バージョン要件**
>
> ML-DSA には OpenSSL 3.5 以降が必要です。古いシステム OpenSSL を使用するプラットフォームでは、カスタムビルドの OpenSSL 3.5+ バイナリを使用する必要があります。

> **RSA キーと EC キーに必要なダイジェストアルゴリズム**
>
> RSA キーと EC キーの場合は、ダイジェストアルゴリズム (など `-sha256`) を指定する必要があります。Ed25519 と ML-DSA にはダイジェストフラグは必要ありません。

```
$ openssl req -provider cloudhsm -provider default -new -sha256 -
key <web_server_fake_pem.key> -out <web_server.csr>
```

3. 自己署名証明書を作成する

次のコマンドを実行して、 AWS CloudHSM OpenSSL プロバイダーを使用して HSM のプライベートキーで CSR に署名します。これにより、自己署名証明書が作成されます。コマンドの以下の値を独自の値に置き換えます。

- *<web_server.csr>* - CSR を含むファイルの名前です。
- *<web_server_fake_pem.key>* - フェイク PEM プライベートキーが含まれるファイルの名前です。
- *<web_server.crt>* - ウェブサーバー証明書が含まれるファイルの名前です。

```
$ openssl x509 -provider cloudhsm -provider default -req -sha256 -days 365 -
in <web_server.csr> -signkey <web_server_fake_pem.key> -out <web_server.crt>
```

プライベートキーと証明書を取得したら、 に移動しますステップ 3: ウェブサーバーを設定する。

ステップ 3: ウェブサーバーを設定する

前のステップで作成した HTTPS 証明書とフェイク PEM プライベートキーを使用するようにウェブサーバーソフトウェアの設定を更新します。開始する前に、既存の証明書とキーを必ずバックアップしてください。これにより、SSL/TLS サーバー ID オフロード用の Linux ウェブサーバーソフトウェアの設定は完了します AWS CloudHSM。

次のいずれかのセクションの手順を完了します。

トピック

- NGINX ウェブサーバーを設定する
- HAProxy ウェブサーバーを設定する

NGINX ウェブサーバーを設定する

このセクションを使用して、OpenSSL プロバイダーで NGINX を設定します。

OpenSSL プロバイダーの NGINX を設定するには

1. クライアントインスタンスに接続します。
2. 次のコマンドを実行して、ウェブサーバー証明書とフェイク PEM プライベートキーに必要なディレクトリを作成します。

```
$ mkdir -p /etc/pki/nginx/private
```

3. 次のコマンドを実行して、ウェブサーバーの証明書を所定場所にコピーします。*<web_server.crt>* を、ウェブサーバー証明書の名前に置き換えます。

```
$ cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. 次のコマンドを実行して、フェイク PEM プライベートキーを所定場所にコピーします。*<web_server_fake_pem.key>* をフェイク PEM プライベートキーが含まれるファイルの名前に置き換えます。

```
$ cp <web_server_fake_pem.key> /etc/pki/nginx/private/server.key
```

5. 次のコマンドを実行してファイルの所有権を変更し、nginx という名前のユーザーがそれらのファイルを読み取れるようにします。

```
$ chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. AWS CloudHSM プロバイダーを使用するように OpenSSL を設定します。OpenSSL プロバイダーの設定の詳細については、「AWS CloudHSM OpenSSL Provider for Client SDK 5」を参照してください。

   1. OpenSSL 設定ファイルを見つけます。

```
$ openssl version -d
```

      次のような出力が表示されます:

```
OPENSSLDIR: "/etc/pki/tls"
```

      設定ファイルは、このディレクトリ`openssl.cnf`にあります。

   2.

> **Note**
>
> システムのデフォルトの openssl.cnf ファイルを直接変更しないでください。これにより、システム全体の OpenSSL オペレーション (SSH、TLS 接続、およびその他のサービス) が CloudHSM プロバイダーを介して意図せずにルーティングされるのを防ぐことができます。
> 別の設定ファイルを使用すると、HSM-backed 暗号化オペレーションを必要とする特定のアプリケーションのみに CloudHSM プロバイダーの使用範囲を設定できます。

      次の内容で新しい OpenSSL 設定ファイルを作成します。

```
$ cat > <example-cloudhsm-openssl.cnf> << 'EOF'
```

```
## NOTE: This should point to the system default openssl config file.
# Replace /etc/pki/tls with the path to your OpenSSL configuration directory
.include </etc/pki/tls>/openssl.cnf

# Override the existing provider_section to include AWS CloudHSM OpenSSL
 Provider as a 3rd party OpenSSL provider
[provider_sect]
default = default_sect
# Include AWS CloudHSM CloudHSM OpenSSL provider
cloudhsm = cloudhsm_sect

[default_sect]
activate = 1

[cloudhsm_sect]
activate = 1
EOF
```

3. CLOUDHSM_PIN 環境変数が Crypto User (CU) 認証情報で設定されていることを確認します。

```
$ export CLOUDHSM_PIN=<username>:<password>
```

4. 更新された設定ファイルを指すようにOPENSSL_CONF環境変数を設定し、プロバイダーがロードされていることを確認します。

```
$ OPENSSL_CONF=/path/to/example-cloudhsm-openssl.cnf openssl list -providers
```

デフォルトプロバイダーと CloudHSM プロバイダーの両方が表示されます。

```
OPENSSL_CONF=/path/to/example-cloudhsm-openssl.cnf openssl list -providers
Providers:
  default
    name: OpenSSL Default Provider
    version: 3.2.2
    status: active
  cloudhsm
    name: AWS CloudHSM OpenSSL Provider
    version: 5.17.0
    status: active
```

7. 次のコマンドを実行して、/etc/nginx/nginx.conf ファイルをバックアップします。

```
$ cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

8. NGINX の設定を更新します。

> **Note**
>
> 各クラスターは、すべての NGINX ウェブサーバーで最大 1,000 の NGINX ワーカープロセスをサポートできます。

Amazon Linux 2023

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには、Linux ルートのアクセス許可が必要になる場合があります。ファイルの先頭に、次の行を追加します。

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters for DHE
 ciphers
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    # ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-
```

```
RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-
POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
    ssl_prefer_server_ciphers off;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

RHEL 9 (9.2+)

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには、Linux ルートのアクセス許可が必要になる場合があります。ファイルの先頭に、次の行を追加します。

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
```

```
    # It is *strongly* recommended to generate unique DH parameters for DHE
 ciphers
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    # ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-
RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-
POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
    ssl_prefer_server_ciphers off;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

RHEL 10 (10.0+)

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには、Linux ルートのアクセス許可が必要になる場合があります。ファイルの先頭に、次の行を追加します。

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters for DHE
 ciphers
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    # ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-
RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-
POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
    ssl_prefer_server_ciphers off;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Ubuntu 26.04 LTS

テキストエディタを使用して、/etc/nginx/nginx.conf ファイルを編集します。これには、Linux ルートのアクセス許可が必要になる場合があります。ファイルの先頭に、次の行を追加します。

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

次に、ファイルの TLS セクションに次の内容を追加します。

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /var/www/html;

    ssl_certificate "/etc/ssl/certs/server.crt";
    ssl_certificate_key "/etc/ssl/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters for DHE
 ciphers
    # Generate them with: openssl dhparam -out /etc/ssl/certs/dhparams.pem 2048
    # ssl_dhparam "/etc/ssl/certs/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-
RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-
POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
    ssl_prefer_server_ciphers off;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }
```

```
    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

ファイルを保存します。

9. systemd 設定ファイルをバックアップしてから、EnvironmentFile パスを設定します。

   Amazon Linux 2023

   1. nginx.service ファイルをバックアップします。

      ```
      $ cp /lib/systemd/system/nginx.service /lib/systemd/system/
      nginx.service.backup
      ```

   2. テキストエディタで /lib/systemd/system/nginx.service を開きます。[サービス] セクションで、以下を追加します。

      ```
      EnvironmentFile=/etc/sysconfig/nginx
      ```

   RHEL 9 (9.2+)

   1. nginx.service ファイルをバックアップします。

      ```
      $ cp /lib/systemd/system/nginx.service /lib/systemd/system/
      nginx.service.backup
      ```

   2. テキストエディタで /lib/systemd/system/nginx.service を開きます。[サービス] セクションで、以下を追加します。

      ```
      EnvironmentFile=/etc/sysconfig/nginx
      ```

RHEL 10 (10.0+)

1. nginx.service ファイルをバックアップします。

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. テキストエディタで /lib/systemd/system/nginx.service を開きます。[サービス] セクションで、以下を追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 26.04 LTS

1. nginx.service ファイルをバックアップします。

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. テキストエディタで /lib/systemd/system/nginx.service を開きます。[サービス] セクションで、以下を追加します。

```
EnvironmentFile=/etc/sysconfig/nginx
```

10. /etc/sysconfig/nginx ファイルの存在を確認してから、次のいずれかを実行します。

- ファイルが存在する場合は、次のコマンドを実行してファイルをバックアップします。

```
$ cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- ファイルが存在しない場合は、テキストエディタを開き、/etc/sysconfig/ フォルダ内に nginx という名前のファイルを作成します。

11. NGINX 環境を設定します。

Amazon Linux 2023

Linux ルートユーザーとして、テキストエディタで /etc/sysconfig/nginx ファイルを開きます。例えば、

```
vi /etc/sysconfig/nginx
```

Cryptography User (CU) 認証情報と OpenSSL 設定ファイルへのパスを追加します。

```
CLOUDHSM_PIN=<CU user name>:<password>
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。*<example-cloudhsm-openssl.cnf> ####*を、 で作成した設定ファイルへのフルパスに置き換えますOpenSSL プロバイダーの NGINX を設定するには。

ファイルを保存します。

RHEL 9 (9.2+)

テキストエディタで /etc/sysconfig/nginx ファイルを開きます。これには、Linux ルートのアクセス許可が必要になる場合があります。Cryptography User (CU) 認証情報と OpenSSL 設定ファイルへのパスを追加します。

```
CLOUDHSM_PIN=<CU user name>:<password>
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。*<example-cloudhsm-openssl.cnf> ####*を、 で作成した設定ファイルへのフルパスに置き換えますOpenSSL プロバイダーの NGINX を設定するには。

ファイルを保存します。

RHEL 10 (10.0+)

テキストエディタで /etc/sysconfig/nginx ファイルを開きます。これには、Linux ルートのアクセス許可が必要になる場合があります。Cryptography User (CU) 認証情報と OpenSSL 設定ファイルへのパスを追加します。

```
CLOUDHSM_PIN=<CU user name>:<password>
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。*<example-cloudhsm-openssl.cnf> ####*を、 で作成した設定ファイルへのフルパスに置き換えますOpenSSL プロバイダーの NGINX を設定するには。

ファイルを保存します。

Ubuntu 26.04 LTS

テキストエディタで /etc/sysconfig/nginx ファイルを開きます。これには、Linux ルートのアクセス許可が必要になる場合があります。Cryptography User (CU) 認証情報と OpenSSL 設定ファイルへのパスを追加します。

```
CLOUDHSM_PIN=<CU user name>:<password>
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

<CU #####> と *<#####>* を CU の認証情報に置き換えます。*<example-cloudhsm-openssl.cnf> ####*を、 で作成した設定ファイルへのフルパスに置き換えますOpenSSL プロバイダーの NGINX を設定するには。

ファイルを保存します。

12. NGINX ウェブサーバーを起動します。

Amazon Linux 2023

すべての NGINX プロセスを停止する

```
$ systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ systemctl daemon-reload
```

NGINX を開始する

```
$ systemctl start nginx
```

RHEL 9 (9.2+)

実行中の NGINX プロセスをすべて停止する

```
$ systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ systemctl start nginx
```

RHEL 10 (10.0+)

実行中の NGINX プロセスをすべて停止する

```
$ systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ systemctl start nginx
```

Ubuntu 26.04 LTS

実行中の NGINX プロセスをすべて停止する

```
$ systemctl stop nginx
```

systemd 設定をリロードして最新の変更を取得する

```
$ systemctl daemon-reload
```

NGINX プロセスを開始する

```
$ systemctl start nginx
```

NGINX を設定したら、 に移動します設定した証明書が HTTPS で使用されていることを検証する。

HAProxy ウェブサーバーを設定する

このセクションを使用して、OpenSSL プロバイダーで HAProxy を設定します。次の例は、CloudHSM 証明書とキーを使用して HAProxy をセットアップする方法を示しています。

OpenSSL プロバイダーの HAProxy を設定するには OpenSSL

1. 既存の結合証明書ファイルが存在する場合は、バックアップします。

```
$ cp server-combined.pem server-combined.pem.backup
```

2. 証明書と CloudHSM フェイク PEM キーを使用して、HAProxy 用の結合証明書ファイルを作成します。

```
$ cat server.crt server.key > server-combined.pem
```

3. 既存の HAProxy 設定をバックアップします。

```
$ cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.backup
```

4. で新しい CloudHSM TLS オフロード設定を作成します/etc/haproxy/haproxy.cfg。

```
global
    daemon
    ssl-provider cloudhsm
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/haproxy/dhparams.pem 2048
    # ssl-dh-param-file /etc/haproxy/dhparams.pem
    ssl-default-bind-ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-
GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-
AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-
SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-CHACHA20-POLY1305
    ssl-default-bind-ciphersuites
 TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
```

```
    ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend haproxy_frontend
    bind *:443 ssl crt /path/to/server-combined.pem
    default_backend web_servers

backend web_servers
    server web1 127.0.0.1:8080 check
```

ファイルの場所と一致するように証明書パスを更新します。

5. HAProxy の環境ファイルを使用するように systemd を設定します。場所は Linux ディストリビューションによって異なります。

   Amazon Linux and RHEL

   HAProxy サービスファイルをバックアップして変更します。

   ```
   $ cp /lib/systemd/system/haproxy.service /lib/systemd/system/
   haproxy.service.backup
   ```

   [サービス] セクションで次の行を編集/lib/systemd/system/haproxy.serviceして追加します。

   ```
   EnvironmentFile=/etc/sysconfig/haproxy
   ```

   Ubuntu

   HAProxy サービスファイルをバックアップして変更します。

   ```
   $ cp /lib/systemd/system/haproxy.service /lib/systemd/system/
   haproxy.service.backup
   ```

   [サービス] セクションで次の行を編集/lib/systemd/system/haproxy.serviceして追加します。

```
EnvironmentFile=/etc/default/haproxy
```

6. システムに適した場所に環境ファイルを作成します。

   Amazon Linux and RHEL

   HAProxy 環境ファイルが存在する場合はバックアップします。

   ```
   $ cp /etc/sysconfig/haproxy /etc/sysconfig/haproxy.backup
   ```

   次の内容/etc/sysconfig/haproxyで HAProxy 環境ファイルを作成します。

   ```
   CLOUDHSM_PIN=<CU user name>:<password>
   ```

   Ubuntu

   HAProxy 環境ファイルが存在する場合はバックアップします。

   ```
   $ cp /etc/default/haproxy /etc/default/haproxy.backup
   ```

   次の内容/etc/default/haproxyで HAProxy 環境ファイルを作成します。

   ```
   CLOUDHSM_PIN=<CU user name>:<password>
   ```

   <CU user name> と *<password>* を CU 認証情報に置き換えます。

7. systemd 設定を再ロードします。

   ```
   $ systemctl daemon-reload
   ```

8. CloudHSM TLS オフロード設定を使用して HAProxy を開始します。

   ```
   $ systemctl start haproxy
   ```

   HAProxy は、カスタム設定ファイルを使用して直接実行することもできます。

   ```
   $ haproxy -f /path/to/haproxy-cloudhsm.cfg
   ```

HAProxy を設定したら、 に移動します設定した証明書が HTTPS で使用されていることを検証する。

ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

で SSL/TLS オフロード用にウェブサーバーを設定したら AWS CloudHSM、インバウンド HTTPS トラフィックを許可するセキュリティグループにウェブサーバーインスタンスを追加します。これにより、ウェブブラウザなどのクライアントがウェブサーバーと HTTPS 接続を確立できるようになります。次に、ウェブサーバーへの HTTPS 接続を行い、SSL/TLS オフロード用に設定した証明書を使用していることを確認します AWS CloudHSM。

トピック

- インバウンド HTTPS 接続の有効化
- 設定した証明書が HTTPS で使用されていることを検証する

インバウンド HTTPS 接続の有効化

クライアント (ウェブブラウザなど) からウェブサーバーに接続するには、インバウンド HTTPS 接続を許可するセキュリティグループを作成します。具体的には、ポート 443 でインバウンドの TCP 接続を許可する必要があります。このセキュリティグループをウェブサーバーに割り当てます。

HTTPS のセキュリティグループを作成してウェブサーバーに割り当てるには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
2. ナビゲーションペインで、[セキュリティグループ] を選択します。
3. [Create Security Group] を選択します。
4. [Create Security Group] で、以下の操作を行います。
    a. [Security group name] に、作成するセキュリティグループの名前を入力します
    b. (オプション) 作成するセキュリティグループの説明を入力します。
    c. [VPC] で、ウェブサーバーのAmazon EC2インスタンスが含まれている VPC を選択します。
    d. [Add rule (ルールの追加)] を選択します。
    e. [タイプ] で、ドロップダウンウィンドウから [HTTPS] を選択します。
    f. [ソース] には、ソースの場所を入力します。
    g. [セキュリティグループの作成] を選択してください。

5. ナビゲーションペインで、[インスタンス] を選択してください。
6. ウェブサーバーインスタンスの横にあるチェックボックスを選択します。
7. ページの上部で [アクション] ドロップダウンメニューを選択します。[セキュリティ] を選択し、[セキュリティグループの変更] を選択します。
8. [関連付けられたセキュリティグループ] で、検索ボックスを選択して HTTPS 用に作成したセキュリティグループを選択します。次に、[セキュリティグループの追加] を選択します。
9. [保存] を選択します。

設定した証明書が HTTPS で使用されていることを検証する

ウェブサーバーをセキュリティグループに追加した後、SSL/TLS オフロードが自己署名証明書を使用していることを確認できます。この検証には、ウェブブラウザ、または OpenSSL s_client などのツールを使用できます。

ウェブブラウザで SSL/TLS オフロードを確認するには

1. ウェブブラウザを使用し、サーバーの公開 DNS 名または IP アドレスを使用してウェブサーバーに接続します。アドレスバーの URL が https:// で始まっていることを確認します。例えば、**`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. ウェブブラウザを使用して、ウェブサーバー証明書を表示します。詳細については次を参照してください:

   - Mozilla Firefox の場合は、Mozilla サポートウェブサイトの「証明書を見る」を参照してください。
   - Google Chrome の場合は、ウェブ開発者向け Google ツールのウェブサイトで「セキュリティの問題を理解する」を参照してください。

   他のウェブブラウザでも、同様の機能を使用してウェブサーバー証明書を表示できる場合があります。

3. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

OpenSSL s_client で SSL/TLS オフロードを確認するには

1. HTTPS を使用してウェブサーバーに接続するには、次の OpenSSL コマンドを実行します。<*#####*> は、ウェブサーバーの公開 DNS 名または IP アドレスに置き換えます。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

これで、ウェブサイトが HTTPS で保護されるようになりました。ウェブサーバーのプライベートキーは、 AWS CloudHSM クラスターの HSM に保存されます。

ロードバランサーを追加するには、「Elastic Load Balancing for でロードバランサーを追加する AWS CloudHSM(オプション)」を参照してください。

AWS CloudHSM JSSE で Tomcat を使用した Linux での SSL/TLS オフロード

このトピックでは、 AWS CloudHSM JCE SDK で Java Secure Socket Extension (JSSE) を使用して SSL/TLS オフロードを設定するstep-by-stepについて説明します。

トピック

- 概要:
- ステップ 1: 前提条件の設定
- ステップ 2: プライベートキーと SSL/TLS 証明書を生成またはインポートする
- ステップ 3: Tomcat ウェブサーバーを設定する
- ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

概要:

では AWS CloudHSM、Tomcat ウェブサーバーは Linux で HTTPS をサポートしています。 AWS CloudHSM JCE SDK には、JSSE (Java Secure Socket Extension) で使用できるインターフェイスが用意されており、このようなウェブサーバーで HSMs を使用できます。 AWS CloudHSM JCE は、JSSE を AWS CloudHSM クラスターに接続するブリッジです。JSSE は、Secure Socket Layer (SSL) と Transport Layer Security (TLS) プロトコル用の Java API です。

ステップ 1: 前提条件の設定

Linux で SSL/TLS オフロード AWS CloudHSM に で Tomcat ウェブサーバーを使用するには、次の前提条件に従います。クライアント SDK 5 と Tomcat ウェブサーバーでウェブサーバー SSL/TLS オフロードを設定するには、これらの前提条件を満たす必要があります。

> **Note**
>
> プラットフォームごとに、異なる前提条件が必要です。使用しているプラットフォームに適したインストール手順を必ず実行してください。

前提条件

- Tomcat ウェブサーバーがインストールされた Linux オペレーティングシステムを実行する Amazon EC2 インスタンス。
- HSM でこのウェブサーバーのプライベートキーを所有および管理する Crypto User (CU)。
- JCE for Client SDK 5 がインストールされ、設定されているハードウェアセキュリティモジュール (HSMs) が少なくとも 2 つあるアクティブな AWS CloudHSM クラスター。 ???

> **Note**
>
> HSM クラスターは1つでも使用できますが、まずクライアントキーの耐久性を無効にする必要があります。詳細については、クライアントキーの耐久性設定の管理 そして クライアント SDK 5 設定ツール を参照してください。

前提条件を満たすには

1. 少なくとも 2 つのハードウェアセキュリティモジュール (HSM) を持つアクティブな AWS CloudHSM クラスター AWS CloudHSM に、 の JCE をインストールして設定します。 HSMs インストールの詳細については、「クライアント SDK 5 向け JCE」を参照してください。
2. AWS CloudHSM クラスターにアクセスできる EC2 Linux インスタンスで、Apache Tomcat の指示に従って Tomcat ウェブサーバーをダウンロードしてインストールします。
3. CloudHSM CLI を使用して Crypto User (CU) を作成します。HSM ユーザーの管理の詳細については、CloudHSM CLI を使用した HSM ユーザー管理について を参照してください。

   > **Tip**
   >
   > CU のユーザー名とパスワードを書き留めます。後に、ウェブサーバーの HTTPS プライベートキーや証明書を生成またはインポートするときに必要になります。

4. Java キーツールを使用して JCE をセットアップするには、クライアント SDK 5 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合する に記載されている手順に従ってください。

以上のステップが完了したら、「ステップ 2: プライベートキーと SSL/TLS 証明書を生成またはインポートする」に進みます。

注意事項

- セキュリティ強化 Linux (SELinux) および Web サーバーを使用するには、クライアント SDK 5 が HSM と通信するために使用するポート 2223 でアウトバウンド TCP 接続を許可する必要があります。
- クラスターを作成してアクティブ化し、EC2 インスタンスにクラスターへのアクセス権を付与するには、AWS CloudHSMの使用開始 の手順を実行します。このセクションでは、1 つの HSM と Amazon EC2 クライアントインスタンスでアクティブなクラスターを作成するためのステップバイステップの手順を提供しています。このクライアントインスタンスをウェブサーバーとして使用することができます。
- クライアントキーの耐久性を無効化しないようにするには、クラスターに複数の HSM を追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。
- クライアントインスタンスに接続するには、SSH または PuTTY を使用することができます。詳細については、「Amazon EC2 ドキュメント」の「SSH を使用した Linux インスタンスへの接続」または「PuTTY を使用した Windows から Linux インスタンスへの接続」を参照してください。

ステップ 2: プライベートキーと SSL/TLS 証明書を生成またはインポートする

HTTPS を有効にするには、Tomcat ウェブサーバーアプリケーションにプライベートキーと、それに対応する SSL/TLS 証明書が必要です。でウェブサーバーの SSL/TLS オフロードを使用するには AWS CloudHSM、プライベートキーを AWS CloudHSM クラスターの HSM に保存する必要があります。

> **Note**
>
> プライベートキーとそれに対応する証明書を持っていない場合、HSM でプライベートキーを生成できます。このプライベートキーを使用して証明書署名リクエスト (CSR) を作成し、それを使用してSSL/TLS証明書を作成します。

HSM のプライベートキーへの参照と関連する証明書を含む local AWS CloudHSM KeyStore ファイルを作成します。ウェブサーバーは、 AWS CloudHSM KeyStore ファイルを使用して、SSL/TLS オフロード中に HSM のプライベートキーを識別します。

トピック

- プライベートキーの生成
- 自己署名証明書を生成します

プライベートキーの生成

このセクションでは、JDK から KeyTool を使用してキーペアを生成する方法について説明します。HSM 内でキーペアを生成したら、それを KeyStore ファイルとしてエクスポートし、それに対応する証明書を生成できます。

ユースケースに応じて、RSA または EC キーペアを生成できます。以下の手順では、RSA キーペアを生成する方法を示します。

KeyTool の **genkeypair** コマンドを使用して RSA キーペアを生成します

1. 下の *<VARIABLES>* を特定のデータに置き換えたら、次のコマンドを使用して `jsse_keystore.keystore` という名前のキーストアファイルを生成します。このファイルには HSM 上のプライベートキーへの参照が含まれます。

```
$ keytool -genkeypair -alias <UNIQUE ALIAS FOR KEYS> -keyalg <KEY ALGORITHM> -
keysize <KEY SIZE> -sigalg <SIGN ALGORITHM> \
```

```
        -keystore <PATH>/<JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -dname CERT_DOMAIN_NAME \
        -J-classpath '-J'$JAVA_LIB'/*:/opt/cloudhsm/java/*:./*' \
        -provider "com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider" \
        -providerpath "$CLOUDHSM_JCE_LOCATION" \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

- ***<PATH>***: キーストアファイルを生成するパス。
- ***<UNIQUE ALIAS FOR KEYS>***: これは HSM 上でキーを一意に識別するために使用されます。このエイリアスは、キーの LABEL 属性として設定されます。
- ***<KEY PASSWORD>***: キーへの参照はローカルキーストアファイルに保存され、このパスワードによってローカル参照が保護されます。
- ***<KEYSTORE PASSWORD>***: これはローカルキーストアファイルのパスワードです。
- ***<JSSE KEYSTORE NAME>***: キーストアファイルの名前。
- ***<CERT DOMAIN NAME>***: X.500 識別名。
- ***<KEY ALGORITHM>***: キーペアを生成するためのキーアルゴリズム (RSA と EC など)。
- ***<KEY SIZE>***: キーペアを生成するためのキーサイズ (たとえば、2,048、3,072、4,096)。
- ***<SIGN ALGORITHM>***: キーペアを生成するためのキーサイズ (たとえば、SHA1withRSA、SHA224withRSA、SHA256withRSA、SHA384withRSA、SHA512withRSA) 。

2. コマンドが成功したことを確認するには、次のコマンドを入力し、RSA キーペアが正常に生成されたことを確認します。

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

自己署名証明書を生成します

キーストアファイルとともにプライベートキーを生成したら、このファイルを使用して証明書署名リクエスト (CSR) と証明書を生成できます。

本稼働環境では、通常、認証局 (CA) を使用して CSR から証明書を作成します。CA は、テスト環境では必要ありません。CA を使用する場合は、CA に CSR ファイルを送信し、HTTPS 用のウェブサーバーで提供される署名付き SSL/TLS 証明書を使用してください。

CA を使用する代わりに、KeyTool を使用して自己署名証明書を作成できます。自己署名証明書はブラウザによって信頼されないため、本稼働環境では使用しないでください。これらは、テスト環境で使用することができます。

> **Warning**
>
> 自己署名証明書はテスト環境でのみ使用する必要があります。本稼働環境では、証明機関を使用して証明書を作成するなど、より安全な方法を使用してください。

証明書を生成する

1. 前のステップで生成したキーストアファイルのコピーを入手します。
2. 次のコマンドを実行して、KeyTool を使用して証明書署名リクエスト (CSR) を作成します。

```
$ keytool -certreq -keyalg RSA -alias unique_alias_for_key -file certreq.csr \
        -keystore <JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

> **Note**
>
> 証明書署名リクエストの出力ファイルは `certreq.csr` です。

証明書に署名する

- 下の *<VARIABLES>* を特定のデータに置き換えた後、次のコマンドを実行して、HSM 上のプライベートキーを使用して CSR に署名します。これにより、自己署名証明書が作成されます。

```
$ keytool -gencert -infile certreq.csr -outfile certificate.crt \
    -alias <UNIQUE ALIAS FOR KEYS> -keypass <KEY_PASSWORD> -
storepass <KEYSTORE_PASSWORD> -sigalg SIG_ALG \
    -storetype CLOUDHSM -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
    -keystore jsse_keystore.keystore
```

> **Note**
>
> `certificate.crt` は、エイリアスのプライベートキーを使用する署名付き証明書です。

Keystore に証明書をインポートする

- 下の *<VARIABLES>* を特定のデータに置き換えた後、次のコマンドを実行して、署名付き証明書を信頼できる証明書としてインポートします。このステップでは、エイリアスで識別されるキーストアエントリに証明書を保存します。

```
$ keytool -import -alias <UNIQUE ALIAS FOR KEYS> -keystore jsse_keystore.keystore \
    -file certificate.crt -storetype CLOUDHSM \
    -v -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
    -keypass <KEY PASSWORD> -storepass <KEYSTORE_PASSWORD>
```

証明書を PEM に変換する

- 次のコマンドを実行して、署名付き証明書ファイル (`.crt`) を PEM に変換します。PEM ファイルは http クライアントからのリクエストの送信に使用されます。

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

これらの手順を完了したら、「ステップ 3: ウェブサーバーの設定」に進みます。

ステップ 3: Tomcat ウェブサーバーを設定する

前のステップで作成した HTTPS 証明書と PEM ファイルを使用するようにウェブサーバーソフトウェアの設定を更新します。開始する前に、既存の証明書とキーを必ずバックアップしてください。これで、 AWS CloudHSMを使用して、Linux ウェブサーバーソフトウェアに SSL/TLS オフロードを設定できます。詳細については、「Apache Tomcat 9 設定リファレンス」を参照してください。

サーバーを停止します

- 下の *<VARIABLES>* を特定のデータに置き換えたら、設定を更新する前に、次のコマンドを実行して Tomcat Server を停止します

```
$ /<TOMCAT DIRECTORY>/bin/shutdown.sh
```

  - ***<TOMCAT DIRECTORY>***: Tomcat のインストールディレクトリ。

Tomcat クラスパスを更新する

1. クライアントインスタンスに接続します。
2. Tomcat インストールフォルダを探します。
3. 以下の *<VARIABLES>* を特定のデータに置き換えたら、次のコマンドを使用して、Tomcat/bin/catalina.sh ファイルclasspathにある Tomcat に Java ライブラリと AWS CloudHSM Java パスを追加します。 Tomcat/bin/catalina

```
$ sed -i 's@CLASSPATH="$CLASSPATH""$CATALINA_HOME"\/bin\/
bootstrap.jar@CLASSPATH="$CLASSPATH""$CATALINA_HOME"\/bin\/bootstrap.jar:'"
        <JAVA LIBRARY>"'\/*:\/opt\/cloudhsm\/java\/*:.\/*@' <TOMCAT PATH> /bin/
catalina.sh
```

- ***<JAVA LIBRARY>***: Java JRE ライブラリの場所。
- ***<TOMCAT PATH>***: Tomcat のインストールフォルダー。

HTTPS コネクタをサーバー設定に追加します。

1. Tomcat のインストールフォルダーに移動します。
2. 下の *<VARIABLES>* を特定のデータに置き換えたら、以下のコマンドを使用して HTTPS コネクタを追加し、前提条件で生成された証明書を使用します。

```
$ sed -i '/<Connector port="8080"/i <Connector port=\"443\" maxThreads=\"200\"
 scheme=\"https\" secure=\"true\" SSLEnabled=\"true\" keystoreType=\"CLOUDHSM\"
 keystoreFile=\"
        <CUSTOM DIRECTORY>/<JSSE KEYSTORE NAME>.keystore\" keystorePass=\"<KEYSTORE
 PASSWORD>\" keyPass=\"<KEY PASSWORD>
        \" keyAlias=\"<UNIQUE ALIAS FOR KEYS>" clientAuth=\"false\" sslProtocol=
\"TLS\"/>' <TOMCAT PATH>/conf/server.xml
```

- ***<CUSTOM DIRECTORY>***: キーストアファイルが置かれているディレクトリ。
- ***<JSSE KEYSTORE NAME>***: キーストアファイルの名前。
- ***<KEYSTORE PASSWORD>***: これはローカルキーストアファイルのパスワードです。
- ***<KEY PASSWORD>***: キーへの参照はローカルキーストアファイルに保存され、このパスワードによってローカル参照が保護されます。
- ***<UNIQUE ALIAS FOR KEYS>***: これは HSM 上でキーを一意に識別するために使用されます。このエイリアスは、キーの LABEL 属性として設定されます。

- ***<TOMCAT PATH>***: Tomcat フォルダへのパス。

サーバーの起動

- 下の *<VARIABLES>* を特定のデータに置き換えたら、以下のコマンドを使用して Tomcat サーバーを起動します。

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

> **Note**
>
> ***<TOMCAT DIRECTORY>*** は、Tomcat のインストールディレクトリの名前です。

ウェブサーバー設定を更新したら、「ステップ 4: HTTPS トラフィックを有効にして証明書を検証する」に移動します。

ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

で SSL/TLS オフロード用にウェブサーバーを設定したら AWS CloudHSM、インバウンド HTTPS トラフィックを許可するセキュリティグループにウェブサーバーインスタンスを追加します。これにより、ウェブブラウザなどのクライアントがウェブサーバーと HTTPS 接続を確立できるようになります。次に、ウェブサーバーに HTTPS 接続し、SSL/TLS オフロード用に設定した証明書を使用していることを確認します AWS CloudHSM。

トピック

- インバウンド HTTPS 接続の有効化
- 設定した証明書が HTTPS で使用されていることを検証する

インバウンド HTTPS 接続の有効化

クライアント (ウェブブラウザなど) からウェブサーバーに接続するには、インバウンド HTTPS 接続を許可するセキュリティグループを作成します。具体的には、ポート 443 でインバウンドの TCP 接続を許可する必要があります。このセキュリティグループをウェブサーバーに割り当てます。

HTTPS のセキュリティグループを作成してウェブサーバーに割り当てるには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。

2. ナビゲーションペインで、[セキュリティグループ] を選択します。
3. [Create Security Group] を選択します。
4. [Create Security Group] で、以下の操作を行います。
   a. [Security group name] に、作成するセキュリティグループの名前を入力します
   b. (オプション) 作成するセキュリティグループの説明を入力します。
   c. [VPC] で、ウェブサーバーのAmazon EC2インスタンスが含まれている VPC を選択します。
   d. [Add rule (ルールの追加)] を選択します。
   e. [タイプ] で、ドロップダウンウィンドウから [HTTPS] を選択します。
   f. [ソース] には、ソースの場所を入力します。
   g. [セキュリティグループの作成] を選択してください。
5. ナビゲーションペインで、[インスタンス] を選択してください。
6. ウェブサーバーインスタンスの横にあるチェックボックスを選択します。
7. ページの上部で [アクション] ドロップダウンメニューを選択します。[セキュリティ] を選択し、[セキュリティグループの変更] を選択します。
8. [関連付けられたセキュリティグループ] で、検索ボックスを選択して HTTPS 用に作成したセキュリティグループを選択します。次に、[セキュリティグループの追加] を選択します。
9. [保存] を選択します。

設定した証明書が HTTPS で使用されていることを検証する

ウェブサーバーをセキュリティグループに追加した後、SSL/TLS オフロードが自己署名証明書を使用していることを確認できます。この検証には、ウェブブラウザ、または OpenSSL s_client などのツールを使用できます。

ウェブブラウザで SSL/TLS オフロードを確認するには

1. ウェブブラウザを使用し、サーバーの公開 DNS 名または IP アドレスを使用してウェブサーバーに接続します。アドレスバーの URL が https:// で始まっていることを確認します。例えば、**`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

> **Tip**
>
> Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. ウェブブラウザを使用して、ウェブサーバー証明書を表示します。詳細については次を参照してください:

   - Mozilla Firefox の場合は、Mozilla サポートウェブサイトの「証明書を見る」を参照してください。
   - Google Chrome の場合は、ウェブ開発者向け Google ツールのウェブサイトで「セキュリティの問題を理解する」を参照してください。

   他のウェブブラウザでも、同様の機能を使用してウェブサーバー証明書を表示できる場合があります。

3. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

OpenSSL s_client で SSL/TLS オフロードを確認するには

1. HTTPS を使用してウェブサーバーに接続するには、次の OpenSSL コマンドを実行します。*<#####>* は、ウェブサーバーの公開 DNS 名または IP アドレスに置き換えます。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

これで、ウェブサイトが HTTPS で保護されるようになりました。ウェブサーバーのプライベートキーは、 AWS CloudHSM クラスターの HSM に保存されます。

ロードバランサーを追加するには、「Elastic Load Balancing for でロードバランサーを追加する AWS CloudHSM(オプション)」を参照してください。

AWS CloudHSM KSP で IIS を使用した Windows での SSL/TLS オフロード

このチュートリアルでは、Windows ウェブサーバーで AWS CloudHSM を使用して SSL/TLS オフロードを設定する手順を説明します。

トピック

- 概要:
- ステップ 1: 前提条件の設定
- ステップ 2: 証明書署名リクエスト (CSR) および証明書を作成する
- ステップ 3: ウェブサーバーを設定する
- ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

概要:

Windows では、Windows Server 用インターネットインフォメーションサービス (IIS) ウェブサーバーアプリケーションは HTTPS をネイティブにサポートしています。Microsoft の Cryptography API:Next Generation (CNG) のAWS CloudHSM キーストレージプロバイダー (KSP) には、暗号化オフロードとキーストレージ用に、IIS がクラスターの HSM を使用することを許可するインターフェイスが用意されています。 AWS CloudHSM KSP は IIS を AWS CloudHSM クラスターに接続するブリッジです。

このチュートリアルでは、以下のことを実行する方法を示します。

- Amazon EC2 インスタンスに、ウェブサーバーソフトウェアをインストールします。
- AWS CloudHSM クラスターに保存されている秘密キーで、HTTPS をサポートするようにウェブサーバソフトウェアを設定します。
- (オプション) Amazon EC2 を使用して 2 台目のウェブサーバーインスタンスを作成し、Elastic Load Balancing を使用してロードバランサーを作成します。ロードバランサーを使用すると、複数のサーバーに負荷を分散することでパフォーマンスを向上させることができます。また、1 つ以上のサーバーに障害が発生した場合、冗長性と高可用性を提供します。

始める準備ができたら、「ステップ 1: 前提条件の設定」を参照してください。

ステップ 1: 前提条件の設定

プラットフォームごとに、異なる前提条件が必要です。以下の前提条件セクションのうち、お使いのプラットフォームに合ったものをご利用ください。

トピック

- クライアント SDK 5 の前提条件
- クライアント SDK 3 の前提条件

クライアント SDK 5 の前提条件

でウェブサーバーの SSL/TLS オフロードを設定するには AWS CloudHSM、以下が必要です。

- 少なくとも 1 つの HSM を持つアクティブな AWS CloudHSM クラスター。
- Windows OS が動作する Amazon EC2 インスタンスで、以下のソフトウェアがインストールされていることを確認します。
  - Windows 用の AWS CloudHSM クライアントソフトウェア。
  - Windows Server 用インターネットインフォメーションサービス (IIS)。
- HSM でこのウェブサーバーのプライベートキーを所有および管理する Crypto User (CU)。

> **Note**
>
> このチュートリアルでは、Microsoft Windows Server 2019 を使用します。Microsoft Windows Server 2016 および 2022 もサポートされています。

Windows Server インスタンスをセットアップし、HSM で CU を作成するには

1. 「開始方法」のステップを完了します。Amazon EC2 クライアントを起動するときは、Windows Server 2019 AMI を選択します。これらのステップを完了すると、少なくとも 1 つの HSM を含むアクティブなクラスターが提供されます。Windows 用のクライアントソフトウェアがインストールされた状態で Windows Server を実行している Amazon EC2 AWS CloudHSM クライアントインスタンスもあります。
2. (オプション) 他の HSM をクラスターに追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。

3. Windows Server に接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
4. CloudHSM CLI を使用して Crypto User (CU) を作成します。CU のユーザー名とパスワードを書き留めます。次のステップを完了するために必要になります。

   > **Note**
   >
   > ユーザーの作成については、「CloudHSM CLI による HSM ユーザーの管理」を参照してください。

5. 前のステップで作成した CU ユーザー名とパスワードを使用して、HSM のログイン認証情報を設定します。
6. ステップ 5 で、Windows 資格情報マネージャーを使用して HSM 認証情報を設定した場合は、SysInternals から `psexec.exe` をダウンロードして、NT Authority\SYSTEM として以下のコマンドを実行します。

   ```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
    --username <USERNAME> --password <PASSWORD>
   ```

   <USERNAME> と *<PASSWORD>* を HSM 認証情報に置き換えてください。

IIS を Windows Server にインストールするには

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. Windows Server で [サーバーマネージャー] を起動します。
3. [サーバー マネージャー] ダッシュボードで、[役割と機能の追加] を選択します。
4. [開始する前に] 情報を読み、[次へ] を選択します。
5. [インストールのタイプ] ページで、[ロールベースまたは機能ベースのインストール] を選択します。次いで、[次へ] を選択します。
6. [サーバーの選択] で、[Select a server from the server pool (サーバープールからサーバーを選択する)] を選択します。次いで、[次へ] を選択します。
7. [Server Roles (サーバーの役割位)] で、以下を実行します。

   a. [Web Server (IIS)] を選択します。

b. [Add features that are required for Web Server (IIS) (Web Server (IIS) に必要な機能を追加する)] で、[機能の追加] を選択します。

c. [次へ] を選択してサーバーロールの選択を完了します。

8. [Features (機能)] を選択し, 、デフォルト設定を使用します。次いで、[次へ] を選択します。
9. [Web Server Role (IIS)] の内容をお読みください。次いで、[次へ] を選択します。
10. [Select role services (ロールサービスの追加)] でデフォルトを受け入れるか、必要に応じて設定を変更します。次いで、[次へ] を選択します。
11. [確認] で確認情報を通読します。次に、[インストール] を選択します。
12. インストールが完了したら、[Close] をクリックします。

以上のステップが完了したら、「ステップ 2: 証明書署名リクエスト (CSR) および証明書を作成する」に進みます。

クライアント SDK 3 の前提条件

でウェブサーバーの SSL/TLS オフロードを設定するには AWS CloudHSM、以下が必要です。

- 少なくとも 1 つの HSM を持つアクティブな AWS CloudHSM クラスター。
- Windows OS が動作する Amazon EC2 インスタンスで、以下のソフトウェアがインストールされていることを確認します。
  - Windows 用の AWS CloudHSM クライアントソフトウェア。
  - Windows Server 用インターネットインフォメーションサービス (IIS)。
- HSM でこのウェブサーバーのプライベートキーを所有および管理する Crypto User (CU)。

> **Note**
>
> このチュートリアルでは Microsoft Windows Server 2016 を使用します。また、Microsoft Windows Server 2012 もサポートされていますが、Microsoft Windows Server 2012 R2 はサポート対象外です。

Windows Server インスタンスをセットアップし、HSM で CU を作成するには

1. 「開始方法」のステップを完了します。Amazon EC2 クライアントを起動する場合は、Windows Server 2016 または Windows Server 2012 AMI を選択します。これらのス

テップを完了すると、少なくとも 1 つの HSM を含むアクティブなクラスターが提供されます。Windows 用のクライアントソフトウェアがインストールされた状態で Windows Server を実行している Amazon EC2 AWS CloudHSM クライアントインスタンスもあります。

2. (オプション) 他の HSM をクラスターに追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。
3. Windows Server に接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
4. CloudHSM CLI を使用して Crypto User (CU) を作成します。CU のユーザー名とパスワードを書き留めます。次のステップを完了するために必要になります。

   > **Note**
   >
   > ユーザーの作成については、「CloudHSM CLI による HSM ユーザーの管理」を参照してください。

5. 前のステップで作成した CU ユーザー名とパスワードを使用して、HSM のログイン認証情報を設定します。
6. ステップ 5 で、Windows 資格情報マネージャーを使用して HSM 認証情報を設定した場合は、SysInternals から `psexec.exe` をダウンロードして、NT Authority\SYSTEM として以下のコマンドを実行します。

   ```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
    --username <USERNAME> --password <PASSWORD>
   ```

   <USERNAME> と *<PASSWORD>* を HSM 認証情報に置き換えてください。

IIS を Windows Server にインストールするには

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. Windows Server で [サーバーマネージャー] を起動します。
3. [サーバー マネージャー] ダッシュボードで、[役割と機能の追加] を選択します。
4. [開始する前に] 情報を読み、[次へ] を選択します。
5. [インストールのタイプ] ページで、[ロールベースまたは機能ベースのインストール] を選択します。次いで、[次へ] を選択します。

6. [サーバーの選択] で、[Select a server from the server pool (サーバープールからサーバーを選択する)] を選択します。次いで、[次へ] を選択します。
7. [Server Roles (サーバーの役割位)] で、以下を実行します。
   a. [Web Server (IIS)] を選択します。
   b. [Add features that are required for Web Server (IIS) (Web Server (IIS) に必要な機能を追加する)] で、[機能の追加] を選択します。
   c. [次へ] を選択してサーバーロールの選択を完了します。
8. [Features (機能)] を選択し, 、デフォルト設定を使用します。次いで、[次へ] を選択します。
9. [Web Server Role (IIS)] の内容をお読みください。次いで、[次へ] を選択します。
10. [Select role services (ロールサービスの追加)] でデフォルトを受け入れるか、必要に応じて設定を変更します。次いで、[次へ] を選択します。
11. [確認] で確認情報を通読します。次に、[インストール] を選択します。
12. インストールが完了したら、[Close] をクリックします。

以上のステップが完了したら、「ステップ 2: 証明書署名リクエスト (CSR) および証明書を作成する」に進みます。

ステップ 2: 証明書署名リクエスト (CSR) および証明書を作成する

HTTPS を有効にするには、SSL/TLS 証明書とそれに対応するプライベートキーがウェブサーバーに必要です。で SSL/TLS オフロードを使用するには AWS CloudHSM、プライベートキーをクラスターの HSM に AWS CloudHSM 保存します。そのためには、Microsoft の Cryptography API:Next Generation (CNG) のAWS CloudHSM キーストレージプロバイダー (KSP) を使用して、証明書署名リクエスト (CSR) を作成します。作成したら、その CSR を認証局 (CA) に送信します。これで、証明書を生成する CSR に署名されます。

トピック

- クライアント SDK 5 を使用して CSR を作成する
- クライアント SDK 3 を使用して CSR を作成する
- 署名証明書を取得してインポートする

クライアント SDK 5 を使用して CSR を作成する

1. Windows Server では、テキストエディタを使用して証明書リクエストファイルを作成し、名前を IISCertRequest.inf とします。IISCertRequest.inf ファイルの内容の例を以下に示します。ファイルで指定可能なセクション、キー、値の詳細については、「Microsoft のドキュメント」を参照してください。値 (ProviderName) は変更しないでください。

```
[Version]
Signature = "$Windows NT$"
[NewRequest]
Subject = "CN=example.com,C=US,ST=Washington,L=Seattle,O=ExampleOrg,OU=WebServer"
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "CloudHSM Key Storage Provider"
KeyUsage = 0xf0
MachineKeySet = True
[EnhancedKeyUsageExtension]
OID=1.3.6.1.5.5.7.3.1
```

2. Windows の certreq コマンドを使用して、前のステップで作成した IISCertRequest.inf ファイルから CSR を作成します。以下の例では、CSR を IISCertRequest.csr という名前のファイルに保存します。証明書リクエストファイルに別のファイル名を使用した場合は、*IISCertRequest.inf* を適切なファイル名に置き換えます。CSR ファイルの *IISCertRequest.csr* は、必要に応じて、別のファイル名に置き換えることができます。

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr

CertReq: Request Created
```

   IISCertRequest.csr ファイルには、CSR が含まれます。署名証明書を取得するには、この CSR が必要です。

クライアント SDK 3 を使用して CSR を作成する

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. クライアント AWS CloudHSM デーモンを起動するには、次のコマンドを使用します。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合。

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. Windows Server では、テキストエディタを使用して証明書リクエストファイルを作成し、名前を IISCertRequest.inf とします。IISCertRequest.inf ファイルの内容の例を以下に示します。ファイルで指定可能なセクション、キー、値の詳細については、「Microsoft のドキュメント」を参照してください。値 (ProviderName) は変更しないでください。

```
[Version]
Signature = "$Windows NT$"
[NewRequest]
Subject = "CN=example.com,C=US,ST=Washington,L=Seattle,O=ExampleOrg,OU=WebServer"
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "Cavium Key Storage Provider"
KeyUsage = 0xf0
MachineKeySet = True
[EnhancedKeyUsageExtension]
OID=1.3.6.1.5.5.7.3.1
```

4. Windows の certreq コマンドを使用して、前のステップで作成した IISCertRequest.inf ファイルから CSR を作成します。以下の例では、CSR を IISCertRequest.csr という名前のファイルに保存します。証明書リクエストファイルに別のファイル名を使用した場合は、*IISCertRequest.inf* を適切なファイル名に置き換えます。CSR ファイルの *IISCertRequest.csr* は、必要に応じて、別のファイル名に置き換えることができます。

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
        SDK Version: 2.03

CertReq: Request Created
```

IISCertRequest.csr ファイルには、CSR が含まれます。署名証明書を取得するには、この CSR が必要です。

署名証明書を取得してインポートする

本稼働環境では、通常、認証局 (CA) を使用して CSR から証明書を作成します。CA は、テスト環境では必要ありません。CA を使用する場合は、CSR ファイル (IISCertRequest.csr) を送信後、その CA を使用して署名済み SSL/TLS 証明書を作成します。

CA を使用する代わりに、OpenSSL のようなツールを使用して、自己署名証明書を作成することもできます。

> ⚠ Warning
>
> 自己署名証明書はブラウザによって信頼されないため、本稼働環境では使用しないでください。これらは、テスト環境で使用することができます。

次の手順では、自己署名証明書を作成してウェブサーバーの CSR に署名する方法を示します。

自己署名証明書を作成するには

1. プライベートキーを作成するには、次の OpenSSL コマンドを使用します。*SelfSignedCA.key* は、必要に応じてプライベートキーを含むファイル名に置き換えることができます。

```
openssl genrsa -aes256 -out SelfSignedCA.key 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.......................................+++
e is 65537 (0x10001)
Enter pass phrase for SelfSignedCA.key:
Verifying - Enter pass phrase for SelfSignedCA.key:
```

2. OpenSSL コマンドを使用して、前のステップで作成したプライベートキーで自己署名証明書を作成します。これは対話型コマンドです。画面の指示を読み、プロンプトに従います。*SelfSignedCA.key* を、プライベートキーを含むファイルの名前に置き換えます (異なる場合)。*SelfSignedCA.crt* は、必要に応じて自己署名証明書を含むファイル名に置き換えることができます。

```
openssl req -new -x509 -days 365 -key SelfSignedCA.key -out SelfSignedCA.crt
Enter pass phrase for SelfSignedCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

自己署名証明書を使用してウェブサーバーの CSR に署名するには

- プライベートキーおよび自己署名証明書を使用して CSR に署名するには、次の OpenSSL コマンドを使用します。以下について、対応データを含むファイルの名前に置き換えます (異なる場合)。
  - *IISCertRequest.csr* - ウェブサーバーの CSR を含むファイルの名前です。
  - *SelfSignedCA.crt* - 自己署名証明書を含むファイルの名前です。
  - *SelfSignedCA.key* - プライベートキーを含むファイルの名前です。
  - *IISCert.crt* - ウェブサーバーの署名証明書を含むファイルの名前です。

```
openssl x509 -req -days 365 -in IISCertRequest.csr \
                            -CA SelfSignedCA.crt \
                            -CAkey SelfSignedCA.key \
                            -CAcreateserial \
                            -out IISCert.crt
Signature ok
subject=/ST=IIS-HSM/L=IIS-HSM/OU=IIS-HSM/O=IIS-HSM/CN=IIS-HSM/C=IIS-HSM
Getting CA Private Key
Enter pass phrase for SelfSignedCA.key:
```

上記のステップが完了したら、ウェブサーバーの証明書 (IISCert.crt) と自己署名証明書 (SelfSignedCA.crt) への署名が完了です。これらのファイルがある場合は、「ステップ 3: ウェブサーバーを設定する」を参照してください。

ステップ 3: ウェブサーバーを設定する

「前のステップ」の最後に作成した HTTPS 証明書を使用するには、IIS ウェブサイトの設定を更新します。これで、 AWS CloudHSMを使用して、SSL/TLS オフロード用に Windows ウェブサーバーソフトウェア (IIS) を設定できます。

自己署名証明書を使用して、CSR に署名した場合は、まずその自己署名証明書を Windows Trusted Root Certification Authorities にインポートする必要があります。

自己署名証明書を Windows Trusted Root Certification Authorities にインポートするには

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. 自己署名証明書を Windows server にコピーします。
3. Windows Server で、[コントロールパネル] を開きます。
4. [Search Control Panel (コントロールパネルを検索)] に **certificates** と入力します。続いて、[Manage computer certificates (コンピュータ証明書の管理)] を選択します。
5. [証明書 (ローカル コンピューター)] の [信頼されたルート証明機関] を展開します。
6. [証明書] を右クリックし、[All Tasks (すべてのタスク)]、[インポート] の順に選択します。
7. [Certificate Import Wizard (証明書インポートウィザード)] で [次へ] を選択します。
8. [Browse (参照)] を選択後、自己署名証明書を検索して選択します。「このチュートリアルの前のステップ」の手順に従って自己署名証明書を作成した場合、自己署名証明書の名前は、`SelfSignedCA.crt` です。開く をクリックします。
9. [次へ] を選択します。
10. [証明書ストア] で、[Place all certificates in the following store (すべての証明書を以下のストアに配置)] を選択します。次に、[Trusted Root Certification Authorities] が [証明書ストア] で選択されていることを確認します。
11. [Next] を選択し、[Finish] を選択します。

IIS ウェブサイトの設定を更新するには

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. AWS CloudHSM クライアントデーモンを起動します。

3. このチュートリアルの前のステップ の最後に作成したウェブサーバーの署名付き証明書を、Windowsサーバーにコピーします。
4. Windows Server では、Windows の certreq コマンドを使用して、以下の例のように署名証明書を受け入れます。*IISCert.crt* をウェブサーバーの署名証明書を含むファイルの名前に置き換えます。

```
C:\>certreq -accept IISCert.crt
        SDK Version: 2.03
```

5. Windows Server で [サーバーマネージャー] を起動します。
6. [Server Manager] ダッシュボードの右上隅で、[ツール]、[Internet Information Services (IIS) Manager] の順に選択します。
7. [Internet Information Services (IIS) Manager] ウィンドウで、サーバー名をダブルクリックします。次に、[Sites (サイト)] をダブルクリックします。ウェブサイトを選択します。
8. [SSL Settings (SSL 設定)] を選択します。ウィンドウの右側の [Bindings (バインディング)] を選択します。
9. [Site Bindings] ウィンドウで、[追加] を選択します。
10. [Type (タイプ)] で、[https] を選択します。[SSL 証明書] で、「このチュートリアルの前のステップ」の最後に作成した HTTPS 証明書を選択します。

> **Note**
>
> 証明書のバインディング中にエラーが発生した場合は、サーバーを再起動し、このステップを再試行します。

11. [OK] を選択してください。

ウェブサイトの設定を更新したら、「ステップ 4: HTTPS トラフィックを有効にして証明書を検証する」に移動します。

ステップ 4: HTTPS トラフィックを有効にして証明書を検証する

で SSL/TLS オフロード用にウェブサーバーを設定したら AWS CloudHSM、インバウンド HTTPS トラフィックを許可するセキュリティグループにウェブサーバーインスタンスを追加します。これにより、ウェブブラウザなどのクライアントがウェブサーバーと HTTPS 接続を確立できるようになります。次に、ウェブサーバーに HTTPS 接続し、SSL/TLS オフロード用に設定した証明書を使用していることを確認します AWS CloudHSM。

トピック

- インバウンド HTTPS 接続の有効化
- 設定した証明書が HTTPS で使用されていることを検証する

インバウンド HTTPS 接続の有効化

クライアント (ウェブブラウザなど) からウェブサーバーに接続するには、インバウンド HTTPS 接続を許可するセキュリティグループを作成します。具体的には、ポート 443 でインバウンドの TCP 接続を許可する必要があります。このセキュリティグループをウェブサーバーに割り当てます。

HTTPS のセキュリティグループを作成してウェブサーバーに割り当てるには

1. Amazon EC2 コンソールの https://console.aws.amazon.com/ec2/ を開いてください。
2. ナビゲーションペインで、[セキュリティグループ] を選択します。
3. [Create Security Group] を選択します。
4. [Create Security Group] で、以下の操作を行います。
    a. [Security group name] に、作成するセキュリティグループの名前を入力します
    b. (オプション) 作成するセキュリティグループの説明を入力します。
    c. [VPC] で、ウェブサーバーのAmazon EC2インスタンスが含まれている VPC を選択します。
    d. [Add rule (ルールの追加)] を選択します。
    e. [タイプ] で、ドロップダウンウィンドウから [HTTPS] を選択します。
    f. [ソース] には、ソースの場所を入力します。
    g. [セキュリティグループの作成] を選択してください。
5. ナビゲーションペインで、[インスタンス] を選択してください。
6. ウェブサーバーインスタンスの横にあるチェックボックスを選択します。
7. ページの上部で [アクション] ドロップダウンメニューを選択します。[セキュリティ] を選択し、[セキュリティグループの変更] を選択します。
8. [関連付けられたセキュリティグループ] で、検索ボックスを選択して HTTPS 用に作成したセキュリティグループを選択します。次に、[セキュリティグループの追加] を選択します。
9. [保存] を選択します。

設定した証明書が HTTPS で使用されていることを検証する

ウェブサーバーをセキュリティグループに追加した後、SSL/TLS オフロードが自己署名証明書を使用していることを確認できます。この検証には、ウェブブラウザ、または OpenSSL s_client などのツールを使用できます。

ウェブブラウザで SSL/TLS オフロードを確認するには

1. ウェブブラウザを使用し、サーバーの公開 DNS 名または IP アドレスを使用してウェブサーバーに接続します。アドレスバーの URL が https:// で始まっていることを確認します。例えば、**`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. ウェブブラウザを使用して、ウェブサーバー証明書を表示します。詳細については次を参照してください:

   - Mozilla Firefox の場合は、Mozilla サポートウェブサイトの「証明書を見る」を参照してください。
   - Google Chrome の場合は、ウェブ開発者向け Google ツールのウェブサイトで「セキュリティの問題を理解する」を参照してください。

   他のウェブブラウザでも、同様の機能を使用してウェブサーバー証明書を表示できる場合があります。

3. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

OpenSSL s_client で SSL/TLS オフロードを確認するには

1. HTTPS を使用してウェブサーバーに接続するには、次の OpenSSL コマンドを実行します。*<#####>* は、ウェブサーバーの公開 DNS 名または IP アドレスに置き換えます。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. SSL/TLS 証明書が、ウェブサーバーに設定したものであることを確認してください。

これで、ウェブサイトが HTTPS で保護されるようになりました。ウェブサーバーのプライベートキーは、 AWS CloudHSM クラスターの HSM に保存されます。

ロードバランサーを追加するには、「Elastic Load Balancing for でロードバランサーを追加する AWS CloudHSM(オプション)」を参照してください。

Elastic Load Balancing for でロードバランサーを追加する AWS CloudHSM(オプション)

1 つのウェブサーバーで SSL/TLS オフロードを設定した後で、さらにウェブサーバーを作成し、HTTPS トラフィックをウェブサーバーにルーティングする Elastic Load Balancing ロードバランサーを作成することができます。ロードバランサーは、2 つ以上のウェブサーバーにトラフィックを分散することで、サーバーに対する負荷を軽減できます。また、ロードバランサーはウェブサーバーのヘルス状態をモニタリングして、正常なサーバーにのみトラフィックをルーティングするため、ウェブサイトの可用性も改善できます。ウェブサーバーに障害が発生すると、ロードバランサーはそのウェブサーバーに対するトラフィックのルーティングを自動的に停止します。

トピック

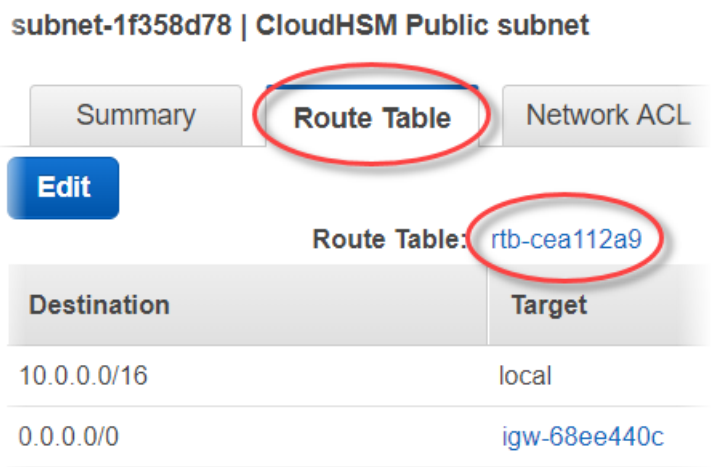
- ステップ 1. 2 番目のウェブサーバーのサブネットを作成する
- ステップ 2. 2 番目のウェブサーバーを作成する
- ステップ 3. ロードバランサーを作成する

ステップ 1. 2 番目のウェブサーバーのサブネットを作成する

別のウェブサーバーを作成する前に、既存のウェブサーバーと AWS CloudHSM クラスターを含む同じ VPC に新しいサブネットを作成する必要があります。

新しいサブネットを作成するには

1. Amazon VPC コンソールの [サブネット] セクションを開きます。
2. [Create Subnet (サブネットの作成)] を選択します。
3. [Create Subnet] ダイアログボックスで、次の操作を行います。
   a. [Name tag] に、サブネットの名前を入力します。
   b. VPC の場合は、既存のウェブサーバーと AWS CloudHSM クラスターを含む AWS CloudHSM VPC を選択します。
   c. [Availability Zone] で、既存のウェブサーバーが含まれているのとは異なるアベイラビリティーゾーンを選択します。
   d. [IPv4 CIDR block] に、サブネットで使用する CIDR ブロックを入力します。例えば、**`10.0.10.0/24`** と入力します。
   e. [はい、作成する] を選択します。
4. 既存のウェブサーバーが含まれているパブリックサブネットの横にあるチェックボックスを選択します。これは、前のステップで作成したパブリックサブネットとは異なります。
5. コンテンツペインで、[ルートテーブル] タブを選択します。次に、ルートテーブルのリンクを選択します。



6. ルートテーブルの横にあるチェックボックスをオンにします。
7. [Subnet Associations] タブを選択します。次に、[編集] を選択します。
8. この手順で前に作成したパブリックサブネットの横にあるチェックボックスを選択します。次に、[Save (保存)] を選択します。

ステップ 2. 2 番目のウェブサーバーを作成する

次の手順を実行し、既存のウェブサーバーと同じ設定で 2 番目のウェブサーバーを作成します。

2 番目のウェブサーバーを作成するには

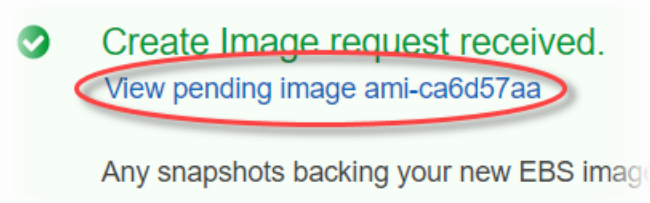
1. Amazon EC2 コンソールの [インスタンス] セクションを開きます。
2. 既存のウェブサーバーインスタンスの横にあるチェックボックスをオンにします。
3. [Actions]、[Image]、[Create Image] の順に選択します。
4. [Create Image] ダイアログボックスで、次の操作を行います。

   a. [Image name] には、イメージの名前を入力します。

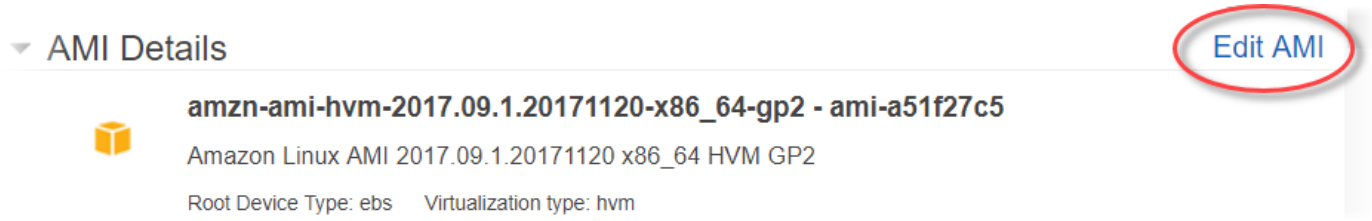
   b. [Image description] としてイメージの説明を入力します。

   c. [Create Image] を選択します。このアクションにより、既存のウェブサーバーが再起動されます。

   d. [View pending image ami-***<AMI ID>***] リンクを選択します。

   [Status] 列で、イメージのステータスを確認します。イメージのステータスが [available] になったら (これには数分かかることがあります)、次のステップに進みます。
5. ナビゲーションペインで、[インスタンス] を選択してください。
6. 既存のウェブサーバーの横にあるチェックボックスをオンにします。
7. [Actions] を選択し、[Launch More Like This] を選択します。
8. [Edit AMI] を選択します。

9. 左側のナビゲーションペインで、[My AMIs] を選択します。次に、検索ボックスのテキストを消去します。
10. ウェブサーバーイメージの横にある [Select] を選択します。

11. [Yes, I want to continue with this AMI (*<image name>* - ami-*<AMI ID>*)] を選択します。
12. [次へ] を選択します。
13. インスタンスタイプを選択し、[次: インスタンスの詳細の設定] を選択します。
14. [ステップ 3: インスタンスの詳細の設定] で、以下の操作を行います。

    a. [Network] で、既存のウェブサーバーが含まれている VPC を選択します。

    b. [Subnet] で、2 番目のウェブサーバー用に作成したパブリックサブネットを選択します。

    c. [Auto-assign Public IP] で、[Enable] を選択します。

    d. 必要に応じて、残りのインスタンスの詳細を変更します。続いて、[次の手順: ストレージの追加] を選択します。

15. 必要に応じて、ストレージの設定を変更します。次に、[次の手順: タグの追加] を選択してください。
16. 必要に応じて、タグを追加または編集します。次に、[次の手順: セキュリティグループの設定] を選択します。
17. [Step 6: Configure Security Group] で、以下の操作を行います。

    a. [セキュリティグループの割り当て] で、[既存のセキュリティグループを選択する] を選択します。

    b. cloudhsm-*<cluster ID>*-SG というという名前のセキュリティグループの横にあるチェックボックスを選択します。 AWS CloudHSM は、クラスターを作成する 際に、代理でこのセキュリティグループを作成しました。ウェブサーバーインスタンスからクラスターの HSM への接続を許可するために、このセキュリティグループを選択する必要があります。

    c. インバウンド HTTPS トラフィックを許可するセキュリティグループの横にあるチェックボックスをオンにします。このセキュリティグループは前に作成済みです。

    d. (オプション) ネットワークからの SSH (Linux) または RDP (Windows) の受信トラフィックを許可するセキュリティ グループの横にあるチェックボックスを選択します。つまり、セキュリティグループは、ポート 22 (Linux の SSH 用) またはポート 3389 (Windows の RDP 用) のインバウンド TCP トラフィックを許可する必要があります。さもないと、クライアントインスタンスに接続することはできません。このようなセキュリティグループがない場合は、作成する必要があります。その後でクライアントインスタンスに割り当てます。

    [Review and Launch] を選択してください。

18. インスタンスの詳細を確認し、[Launch] を選択します。

19. インスタンスを起動するために、既存のキーペアを使用するか、新しいキーペアを作成するか、キーペアを使用しないかを選択します。

    - 既存のキーペアを使用するには、以下の操作を行います。
      1. [Choose an existing key pair] (既存のキーペアの選択) をクリックします。
      2. [Select a key pair] で、使用するキーペアを選択します。
      3. [I acknowledge that I have access to the selected private key file (***<private key file name>***.pem), and that I will not be able to log in my instance.] の横のチェックボックスを選択します。
    - 新しいキーペアを作成するには、以下の操作を行います。
      1. 新規キーペア作成を選択します。
      2. [Key pair name] にキーペアの名前を入力します。
      3. [Download Key Pair] を選択して、プライベートキーファイルを安全でアクセス可能な場所に保存します。

      > ⚠ Warning
      >
      > この時点以降、プライベートキーファイルをダウンロードすることはできません。この時点でプライベートキーファイルをダウンロードしないと、以後はクライアントインスタンスにアクセスできなくなります。

    - キーペアを使用しないでインスタンスを起動するには、次の操作を行います。
      1. [Proceed without a key pair] を選択します。
      2. [I acknowledge that I will not be able to connect to this instance unless I already know the password built into this AMI] の横にあるチェックボックスをオンにします。

    [Launch Instances] (インスタンスを起動) をクリックします。

ステップ 3. ロードバランサーを作成する

以下の手順を完了し、HTTPS トラフィックをウェブサーバーにルーティングする Elastic Load Balancing ロードバランサを作成します。

ロードバランサーを作成するには

1. Amazon EC2 コンソールで [ロードバランサー] セクションを開きます。

2. [Create Load Balancer] を選択します。
3. [Network Load Balancer] セクションで、[Create] を選択します。
4. [Step 1: Configure Load Balancer] で、以下の操作を行います。

    a. [Name] に、作成するロードバランサーの名前を入力します。
    b. [Listeners] セクションの [Load Balancer Port] で、値を **443** に変更します。
    c. [Availability Zones] セクションの [VPC] で、ウェブサーバーが含まれている VPC を選択します。
    d. [Availability Zones] セクションで、ウェブサーバーが含まれているサブネットを選択します。
    e. [Next: Configure Routing] を選択します。
5. [Step 2: Configure Routing] で、以下の操作を行います。

    a. [Name] に、作成するターゲットグループの名前を入力します。
    b. [Port] で、値を **443** に変更します。
    c. [Next: Register Targets] を選択します。
6. [Step 3: Register Targets] で、次の操作を行ないます。

    a. [Instances] セクションで、ウェブサーバーインスタンスの横にあるチェックボックスを選択します。次に、[Add to registered] を選択します。
    b. [次へ: レビュー] を選択します。
7. ロードバランサーの詳細を確認し、[Create] を選択します。
8. ロードバランサーが正常に作成されたら、[Close] を選択します。

上記の手順を完了すると、Amazon EC2 コンソールに Elastic Load Balancing ロードバランサが表示されます。

ロードバランサーの状態がアクティブになると、ロードバランサーが動作していることを確認できます。つまり、 AWS CloudHSMでの SSL/TLS オフロードを使用して HTTPS トラフィックがウェブサーバーに送信されていることを検証できます。これは、ウェブブラウザや OpenSSL s_client などのツールを使用して行うことができます。

ロードバランサーが動作していることをウェブブラウザで確認するには

1. Amazon EC2 コンソールで、先ほど作成したロードバランサーの DNS 名 を見つけてください。次に、この DNS 名を選択してコピーします。

2. Mozilla Firefox や Google Chrome などのウェブブラウザで、ロードバランサーの DNS 名を使用してロードバランサーに接続します。アドレスバーの URL が https:// で始まっていることを確認します。

   > **Tip**
   >
   > Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

3. ウェブブラウザを使用して、ウェブサーバー証明書を表示します。詳細については次を参照してください:

   - Mozilla Firefox の場合は、Mozilla サポートウェブサイトの「証明書を見る」を参照してください。
   - Google Chrome の場合は、ウェブ開発者向け Google ツールのウェブサイトで「セキュリティの問題を理解する」を参照してください。

   他のウェブブラウザでも、同様の機能を使用してウェブサーバー証明書を表示できる場合があります。

4. 証明書が、ウェブサーバーで使用するように設定したものであることを確認します。

ロードバランサーが動作していることを OpenSSL s_client で確認するには

1. 以下の OpenSSL コマンドにより、HTTPS を使用してロードバランサーに接続します。*<DNS name>* を、使用しているロードバランサーの DNS 名に置き換えます。

   ```
   openssl s_client -connect <DNS name>:443
   ```

   > **Tip**
   >
   > Amazon Route 53 などの DNS サービスを使用して、ウェブサイトのドメイン名 (https://www.example.com/ など) をウェブサーバーにルーティングできます。詳細については、Amazon Route 53 開発者ガイド の Amazon EC2 インスタンスへのトラフィックのルーティング または DNS サービスのドキュメントを参照してください。

2. 証明書が、ウェブサーバーで使用するように設定したものであることを確認します。

これで、HTTPS で保護されたウェブサイトが作成され、ウェブサーバーのプライベートキーが AWS CloudHSM クラスターの HSM に保存されました。ウェブサイトは、2 つのウェブサーバーとロードバランサーにより、効率と可用性が向上します。

で認証機関 (CA) として Windows Server を設定する AWS CloudHSM

AWS CloudHSM は、クライアント SDK 3 およびクライアント SDK 5 を介して Windows Server を認証機関 (CA) として設定するサポートを提供します。これらのツールを使用する手順は、現在ダウンロードしているクライアント SDK のバージョンによって異なります。次のセクションでは、各 SDK に関する情報を提供します。

トピック

- Client SDK 5 を使用した認証局 (CA) として Windows Server を設定します。
- Client SDK 3 を使用した認証局 (CA) として Windows Server を設定します。

Client SDK 5 を使用した認証局 (CA) として Windows Server を設定します。

公開鍵基盤 (PKI) において、認証局 (CA) は、デジタル証明書を発行する信頼されたエンティティです。これらのデジタル証明書は、公開鍵暗号方式およびデジタル署名を使用して、パブリックキーを ID (個人または組織) にバインドします。CA を操作するには、CA によって発行された証明書に署名するプライベートキーを保護して、信頼関係を維持する必要があります。プライベートキーを AWS CloudHSM クラスター内の HSM に保存し、HSM を使用して暗号化署名オペレーションを実行できます。

このチュートリアルでは、Windows Server と AWS CloudHSM を使用して CA を設定します。Windows 用 AWS CloudHSM クライアントソフトウェアを Windows Server にインストールしたら、Windows Server に Active Directory Certificate Services (AD CS) のロールを追加します。このロールを設定するときは、 AWS CloudHSM キーストレージプロバイダー (KSP) を使用して CA のプライベートキーを作成し、 AWS CloudHSM クラスターに保存します。KSP は、Windows サーバーを AWS CloudHSM クラスターに接続するブリッジです。最後のステップで、Windows Server CA を使用して、証明書署名リクエスト (CSR) に署名します。

詳細については、以下の各トピックを参照してください。

トピック

- ステップ 1: 前提条件の設定
- ステップ 2: を使用して Windows Server CA を作成する AWS CloudHSM
- ステップ 3: を使用して Windows Server CA で証明書署名リクエスト (CSR) に署名する AWS CloudHSM

ステップ 1: 前提条件の設定

で Windows Server を認証機関 (CA) としてセットアップするには AWS CloudHSM、以下が必要です。

- 少なくとも 1 つの HSM を持つアクティブな AWS CloudHSM クラスター。
- Windows 用の AWS CloudHSM クライアントソフトウェアがインストールされた Windows Server オペレーティングシステムを実行する Amazon EC2 インスタンス。このチュートリアルでは Microsoft Windows Server 2016 を使用します。
- HSM で CA のプライベートキーを所有および管理するための暗号化ユーザー (CU)。

を使用して Windows Server CA の前提条件を設定するには AWS CloudHSM

1. 「開始方法」のステップを完了します。Amazon EC2 クライアントを起動するときは、Windows Server AMIを選択します。このチュートリアルでは Microsoft Windows Server 2016 を使用します。これらのステップを完了すると、少なくとも 1 つの HSM を含むアクティブなクラスターが提供されます。Windows 用のクライアントソフトウェアがインストールされた状態で Windows Server を実行している Amazon EC2 AWS CloudHSM クライアントインスタンスもあります。
2. (オプション) 他の HSM をクラスターに追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。
3. クライアントインスタンスに接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
4. Managing HSM users with CloudHSM CLI または Managing HSM users with CloudHSM Management Utility (CMU) を使用して、Crypto User (CU) を作成します。CU のユーザー名とパスワードを書き留めます。次のステップを完了するために必要になります。

5. 前のステップで作成した CU ユーザー名とパスワードを使用して、HSM のログイン認証情報を設定します。
6. ステップ 5 で、Windows 資格情報マネージャーを使用して HSM 認証情報を設定した場合は、SysInternals から psexec.exe をダウンロードして、NT Authority\SYSTEM として以下のコマンドを実行します。

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
 --username <USERNAME> --password <PASSWORD>
```

<USERNAME> と *<PASSWORD>* を HSM 認証情報に置き換えてください。

を使用して Windows Server CA を作成するには AWS CloudHSM、「」を参照してくださいWindows Server CA の作成。

ステップ 2: を使用して Windows Server CA を作成する AWS CloudHSM

Windows Server CA を作成するには、Active Directory 証明書サービス (AD CS) ロールを Windows Server に追加します。このロールを追加するときは、 AWS CloudHSM キーストレージプロバイダー (KSP) を使用して CA のプライベートキーを作成し、 AWS CloudHSM クラスターに保存します。

> **Note**
>
> Windows Server CA を作成すると、ルート CA または下位 CA の作成を選択できます。通常、この決定はパブリックキーインフラストラクチャの設計および組織のセキュリティポリシーに基づいて行います。このチュートリアルではルート CA を作成する方法を簡単に説明します。

AD CS ロールを Windows Server に追加して CA のプライベートキーを作成するには

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. Windows Server で [サーバーマネージャー] を起動します。
3. [サーバー マネージャー] ダッシュボードで、[役割と機能の追加] を選択します。
4. [開始する前に] 情報を読み、[次へ] を選択します。

5. [インストールのタイプ] ページで、[ロールベースまたは機能ベースのインストール] を選択します。次いで、[次へ] を選択します。
6. [サーバーの選択] で、[Select a server from the server pool (サーバープールからサーバーを選択する)] を選択します。次いで、[次へ] を選択します。
7. [Server Roles (サーバーの役割位)] で、以下を実行します。
   a. [Active Directory Certificate Services] を選択します。
   b. [Add features that are required for Active Directory Certificate Services (Active Directory Certificate Services に必要な機能を追加する)] で、[機能の追加] を選択します。
   c. [次へ] を選択してサーバーロールの選択を完了します。
8. [機能] で、デフォルトを受け入れて、[次へ] を選択します。
9. [AD CS] で、以下を実行します
   a. [次へ] を選択します。
   b. [証明機関] を選択してから、[次へ] を選択します。
10. [確認] で確認情報を読み、[インストール] を選択します。ウィンドウを閉じないでください。
11. 強調表示された [Configure Active Directory Certificate Services on the destination server (ターゲットサーバーの Active Directory Certificate Services を設定する)] リンクを選択します。
12. [認証情報] で、表示される認証情報を検証または変更します。次いで、[次へ] を選択します。
13. [役割サービス] で、[証明機関] を選択します。次いで、[次へ] を選択します。
14. [Setup Type (セットアップタイプ)] で、[Standalone CA (スタンドアロン CA)] を選択します。次いで、[次へ] を選択します。
15. [CA Type (CA タイプ)] で、[Root CA (ルート CA)] を選択します。次いで、[次へ] を選択します。

> **Note**
>
> パブリックキーインフラストラクチャの設計と組織のセキュリティポリシーに基づいて、ルート CA または下位 CA の作成を選択できます。このチュートリアルではルート CA を作成する方法を簡単に説明します。

16. [プライベートキー] で、[Create a new private key (新しいプライベートキーを作成する)] を選択します。次いで、[次へ] を選択します。
17. [暗号化] で、以下の操作を実行します。

a. [暗号化プロバイダーを選択する] で、メニューから [CloudHSM キーストレージプロバイダー] のいずれかを選択します。これらは、 AWS CloudHSM キーストレージプロバイダーです。たとえば、[RSA#CloudHSM キーストレージプロバイダー] を選択できます。

b. [Key length (キーの長さ)] で、キーの長さのオプションを 1 つ選択します。

c. [Select the hash algorithm for signing certificates issued by this CA (この CA によって発行された証明書に署名するためのハッシュアルゴリズムを選択する)] で、ハッシュアルゴリズムのオプションを 1 つ選択します。

[次へ] を選択します。

18. [CA Name (CA 名)] で、以下を実行します。

a. (省略可能) 共通名を編集します。

b. (省略可能) 識別子名サフィックスを入力します。

[次へ] を選択します。

19. [有効期間] で、期間を年、月、週、または日で指定します。次いで、[次へ] を選択します。

20. [Certificate Database (証明書データベース)] では、デフォルト値をそのままにするか、必要に応じてデータベースとデータベースログの場所を変更できます。次いで、[次へ] を選択します。

21. [確認] で、CA に関する情報を確認し、[構成する] を選択します。

22. [閉じる] を選択し、もう一度 [閉じる] を選択します。

これで、 で Windows Server CA が作成されました AWS CloudHSM。CA を使用して証明書署名リクエスト (CSR) に署名する方法の詳細については、「CSR への署名」を参照してください。

ステップ 3: を使用して Windows Server CA で証明書署名リクエスト (CSR) に署名する AWS CloudHSM

で Windows Server CA を使用して AWS CloudHSM 、証明書署名リクエスト (CSR) に署名できます。これらのステップを完了するには、有効な CSR が必要です。CSR は以下を含むいくつかの方法で作成できます。

- OpenSSL の使用
- Windows Server インターネット インフォメーション サービス (IIS) マネージャを使用する
- Microsoft マネジメントコンソールの証明書スナップインを使用する

- Windows で certreq コマンドラインユーティリティを使用する

CSR を作成する手順は、このチュートリアルの範囲外です。CSR がある場合は、Windows Server CA を使用して署名できます。

Windows Server CA を使用して CSR に署名するには

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. Windows Server で [サーバーマネージャー] を起動します。
3. [サーバー マネージャー] ダッシュボードの右上隅で、[ツール]、[証明機関] の順に選択します。
4. [証明機関] ウィンドウで、コンピュータ名を選択します。
5. [アクション] メニューで、[すべてのタスク]、[新しい要求の送信] の順に選択します。
6. CSR ファイルを選択して、[開く] を選択します。
7. [証明機関] ウィンドウで、[保留中の要求] をダブルクリックします。
8. 保留中のリクエストを選択します。次に、[アクション] メニューで、[すべてのタスク]、[発行] の順に選択します。
9. [証明機関] ウィンドウで、[Issued Requests (発行済みの要求)] をダブルクリックします。
10. (オプション) 署名付き証明書をファイルにエクスポートするには、次のステップを実行します。
    a. [証明機関] ウィンドウで、証明書をダブルクリックします。
    b. [詳細] タブ、[Copy to File (ファイルにコピー)] の順に選択します。
    c. [Certificate Export Wizard (証明書のエクスポートウィザード)] の手順に従います。

これで、 を使用する Windows Server CA と AWS CloudHSM、Windows Server CA によって署名された有効な証明書が作成されました。

Client SDK 3 を使用した認証局 (CA) として Windows Server を設定します。

公開鍵基盤 (PKI) において、認証局 (CA) は、デジタル証明書を発行する信頼されたエンティティです。これらのデジタル証明書は、公開鍵暗号方式およびデジタル署名を使用して、パブリックキーを ID (個人または組織) にバインドします。CA を操作するには、CA によって発行された証明書に署名するプライベートキーを保護して、信頼関係を維持する必要があります。プライベートキーを AWS

CloudHSM クラスター内の HSM に保存し、HSM を使用して暗号化署名オペレーションを実行できます。

このチュートリアルでは、Windows Server と AWS CloudHSM を使用して CA を設定します。Windows 用 AWS CloudHSM クライアントソフトウェアを Windows Server にインストールしたら、Windows Server に Active Directory Certificate Services (AD CS) のロールを追加します。このロールを設定するときは、 AWS CloudHSM キーストレージプロバイダー (KSP) を使用して CA のプライベートキーを作成し、 AWS CloudHSM クラスターに保存します。KSP は、Windows サーバーを AWS CloudHSM クラスターに接続するブリッジです。最後のステップで、Windows Server CA を使用して、証明書署名リクエスト (CSR) に署名します。

詳細については、以下の各トピックを参照してください。

トピック

- ステップ 1: 前提条件の設定
- ステップ 2: を使用して Windows Server CA を作成する AWS CloudHSM
- ステップ 3: を使用して Windows Server CA で証明書署名リクエスト (CSR) に署名する AWS CloudHSM

ステップ 1: 前提条件の設定

で Windows Server を認証機関 (CA) としてセットアップするには AWS CloudHSM、以下が必要です。

- 少なくとも 1 つの HSM を持つアクティブな AWS CloudHSM クラスター。
- Windows 用の AWS CloudHSM クライアントソフトウェアがインストールされた Windows Server オペレーティングシステムを実行する Amazon EC2 インスタンス。このチュートリアルでは Microsoft Windows Server 2016 を使用します。
- HSM で CA のプライベートキーを所有および管理するための暗号化ユーザー (CU)。

を使用して Windows Server CA の前提条件を設定するには AWS CloudHSM

1. 「開始方法」のステップを完了します。Amazon EC2 クライアントを起動するときは、Windows Server AMIを選択します。このチュートリアルでは Microsoft Windows Server 2016 を使用します。これらのステップを完了すると、少なくとも 1 つの HSM を含むアクティブなクラスターが提供されます。Windows 用のクライアントソフトウェアがインストールされ

た状態で Windows Server を実行している Amazon EC2 AWS CloudHSM クライアントインスタンスもあります。

2. (オプション) 他の HSM をクラスターに追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。
3. クライアントインスタンスに接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
4. Managing HSM users with CloudHSM CLI または Managing HSM users with CloudHSM Management Utility (CMU) を使用して、Crypto User (CU) を作成します。CU のユーザー名とパスワードを書き留めます。次のステップを完了するために必要になります。
5. 前のステップで作成した CU ユーザー名とパスワードを使用して、HSM のログイン認証情報を設定します。
6. ステップ 5 で、Windows 資格情報マネージャーを使用して HSM 認証情報を設定した場合は、SysInternals から `psexec.exe` をダウンロードして、NT Authority\SYSTEM として以下のコマンドを実行します。

```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools
\set_cloudhsm_credentials.exe" --username <USERNAME> --password <PASSWORD>
```

<USERNAME> と *<PASSWORD>* を HSM 認証情報に置き換えてください。

を使用して Windows Server CA を作成するには AWS CloudHSM、「」を参照してくださいWindows Server CA の作成。

ステップ 2: を使用して Windows Server CA を作成する AWS CloudHSM

Windows Server CA を作成するには、Active Directory 証明書サービス (AD CS) ロールを Windows Server に追加します。このロールを追加するときは、 AWS CloudHSM キーストレージプロバイダー (KSP) を使用して CA のプライベートキーを作成し、 AWS CloudHSM クラスターに保存します。

> **Note**
>
> Windows Server CA を作成すると、ルート CA または下位 CA の作成を選択できます。通常、この決定はパブリックキーインフラストラクチャの設計および組織のセキュリティポリ

シーに基づいて行います。このチュートリアルではルート CA を作成する方法を簡単に説明します。

AD CS ロールを Windows Server に追加して CA のプライベートキーを作成するには

1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. Windows Server で [サーバーマネージャー] を起動します。
3. [サーバー マネージャー] ダッシュボードで、[役割と機能の追加] を選択します。
4. [開始する前に] 情報を読み、[次へ] を選択します。
5. [インストールのタイプ] ページで、[ロールベースまたは機能ベースのインストール] を選択します。次いで、[次へ] を選択します。
6. [サーバーの選択] で、[Select a server from the server pool (サーバープールからサーバーを選択する)] を選択します。次いで、[次へ] を選択します。
7. [Server Roles (サーバーの役割位)] で、以下を実行します。
   a. [Active Directory Certificate Services] を選択します。
   b. [Add features that are required for Active Directory Certificate Services (Active Directory Certificate Services に必要な機能を追加する)] で、[機能の追加] を選択します。
   c. [次へ] を選択してサーバーロールの選択を完了します。
8. [機能] で、デフォルトを受け入れて、[次へ] を選択します。
9. [AD CS] で、以下を実行します
   a. [次へ] を選択します。
   b. [証明機関] を選択してから、[次へ] を選択します。
10. [確認] で確認情報を読み、[インストール] を選択します。ウィンドウを閉じないでください。
11. 強調表示された [Configure Active Directory Certificate Services on the destination server (ターゲットサーバーの Active Directory Certificate Services を設定する)] リンクを選択します。
12. [認証情報] で、表示される認証情報を検証または変更します。次いで、[次へ] を選択します。
13. [役割サービス] で、[証明機関] を選択します。次いで、[次へ] を選択します。
14. [Setup Type (セットアップタイプ)] で、[Standalone CA (スタンドアロン CA)] を選択します。次いで、[次へ] を選択します。

15. [CA Type (CA タイプ)] で、[Root CA (ルート CA)] を選択します。次いで、[次へ] を選択します。

    > **Note**
    >
    > パブリックキーインフラストラクチャの設計と組織のセキュリティポリシーに基づいて、ルート CA または下位 CA の作成を選択できます。このチュートリアルではルート CA を作成する方法を簡単に説明します。

16. [プライベートキー] で、[Create a new private key (新しいプライベートキーを作成する)] を選択します。次いで、[次へ] を選択します。
17. [暗号化] で、以下の操作を実行します。

    a. [Select a cryptographic provider (暗号化プロバイダーを選択する)] で、メニューから [Cavium Key Storage Provider (Cavium キーストレージプロバイダー)] のいずれかを選択します。これらは、 AWS CloudHSM キーストレージプロバイダーです。たとえば、[RSA#Cavium Key Storage Provider (RSA#Cavium キーストレージプロバイダー)] を選択できます。

    b. [Key length (キーの長さ)] で、キーの長さのオプションを 1 つ選択します。

    c. [Select the hash algorithm for signing certificates issued by this CA (この CA によって発行された証明書に署名するためのハッシュアルゴリズムを選択する)] で、ハッシュアルゴリズムのオプションを 1 つ選択します。

    [次へ] を選択します。
18. [CA Name (CA 名)] で、以下を実行します。

    a. (省略可能) 共通名を編集します。

    b. (省略可能) 識別子名サフィックスを入力します。

    [次へ] を選択します。
19. [有効期間] で、期間を年、月、週、または日で指定します。次いで、[次へ] を選択します。
20. [Certificate Database (証明書データベース)] では、デフォルト値をそのままにするか、必要に応じてデータベースとデータベースログの場所を変更できます。次いで、[次へ] を選択します。
21. [確認] で、CA に関する情報を確認し、[構成する] を選択します。
22. [閉じる] を選択し、もう一度 [閉じる] を選択します。

これで、 で Windows Server CA が作成されました AWS CloudHSM。CA を使用して証明書署名リクエスト (CSR) に署名する方法の詳細については、「CSR への署名」を参照してください。

ステップ 3: を使用して Windows Server CA で証明書署名リクエスト (CSR) に署名する AWS CloudHSM

で Windows Server CA を使用して AWS CloudHSM 、証明書署名リクエスト (CSR) に署名できます。これらのステップを完了するには、有効な CSR が必要です。CSR は以下を含むいくつかの方法で作成できます。

- OpenSSL の使用
- Windows Server インターネット インフォメーション サービス (IIS) マネージャを使用する
- Microsoft マネジメントコンソールの証明書スナップインを使用する
- Windows で certreq コマンドラインユーティリティを使用する

CSR を作成する手順は、このチュートリアルの範囲外です。CSR がある場合は、Windows Server CA を使用して署名できます。

Windows Server CA を使用して CSR に署名するには

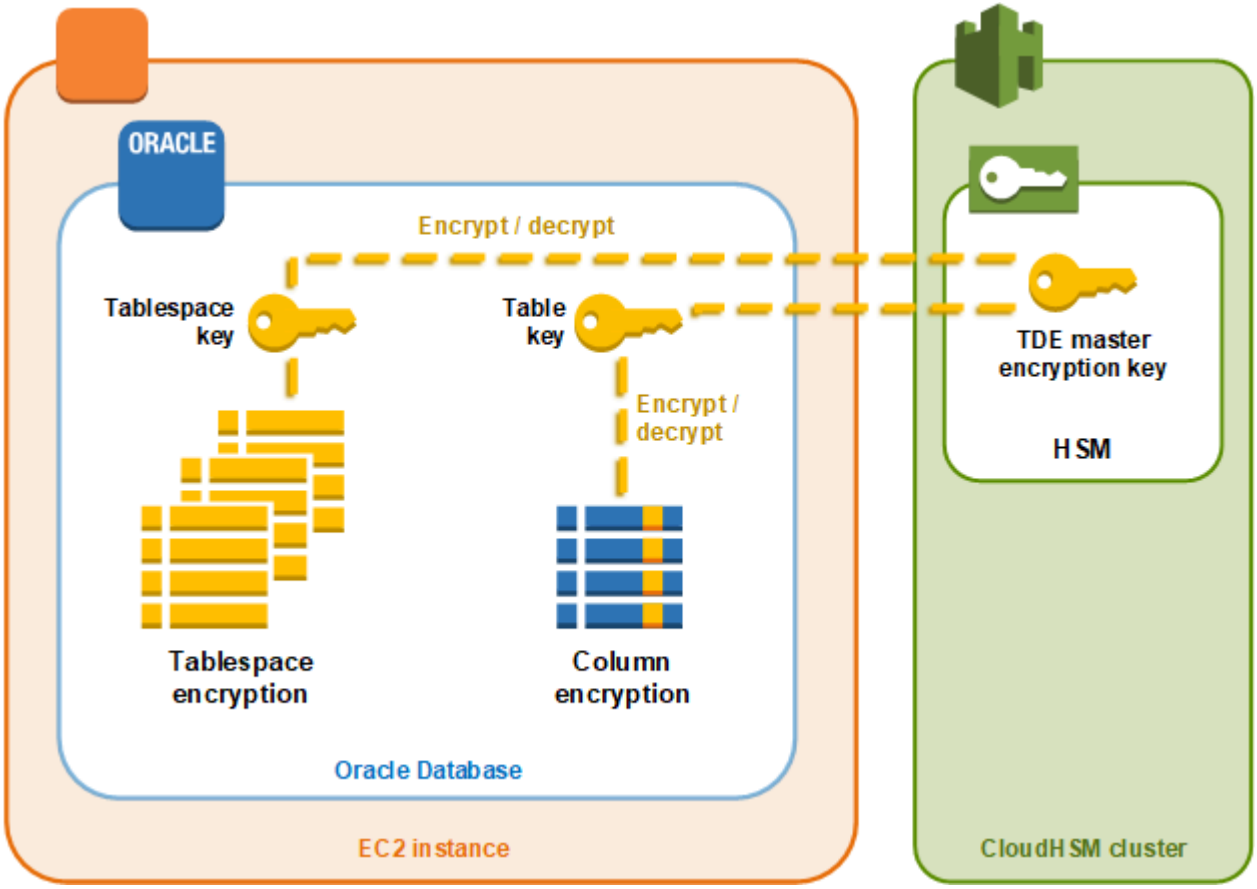
1. Windows Server に接続していない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. Windows Server で [サーバーマネージャー] を起動します。
3. [サーバー マネージャー] ダッシュボードの右上隅で、[ツール]、[証明機関] の順に選択します。
4. [証明機関] ウィンドウで、コンピュータ名を選択します。
5. [アクション] メニューで、[すべてのタスク]、[新しい要求の送信] の順に選択します。
6. CSR ファイルを選択して、[開く] を選択します。
7. [証明機関] ウィンドウで、[保留中の要求] をダブルクリックします。
8. 保留中のリクエストを選択します。次に、[アクション] メニューで、[すべてのタスク]、[発行] の順に選択します。
9. [証明機関] ウィンドウで、[Issued Requests (発行済みの要求)] をダブルクリックします。
10. (オプション) 署名付き証明書をファイルにエクスポートするには、次のステップを実行します。
    a. [証明機関] ウィンドウで、証明書をダブルクリックします。
    b. [詳細] タブ、[Copy to File (ファイルにコピー)] の順に選択します。

c. [Certificate Export Wizard (証明書のエクスポートウィザード)] の手順に従います。

これで、 を使用する Windows Server CA と AWS CloudHSM、Windows Server CA によって署名された有効な証明書が作成されました。

を使用した Oracle データベース透過的なデータ暗号化 (TDE) AWS CloudHSM

透過的なデータ暗号化 (TDE) を使用して、データベースファイルを暗号化します。TDE を使用すると、データベースソフトウェアはデータをディスクに保存する前に暗号化します。データベースのテーブル列またはテーブルスペースのデータは、テーブルキーまたはテーブルスペースキーで暗号化されています。一部バージョンの Oracle のデータベースソフトウェアには TDE を提供します。Oracle TDE では、これらのキーは TDE マスター暗号化キーを使用して暗号化されます。TDE マスター暗号化キーを AWS CloudHSM クラスターの HSMs に保存することで、セキュリティを強化できます。



このソリューションでは、Amazon EC2 インスタンスにインストールされている Oracle Database を使用します。Oracle Database は PKCS #11 のAWS CloudHSM ソフトウェアライブラリと統合し、TDE マスターキーをクラスター内の HSM に保存します。

> **Important**
>
> - Amazon EC2 インスタンスに Oracle Database をインストールすることを推奨します。

Oracle TDE と AWS CloudHSMを統合するには、次のステップを実行します。

トピック

- ステップ 1. 前提条件の設定
- ステップ 2: Oracle データベース設定を更新する
- ステップ 3: Oracle TDE マスター暗号化キーの生成

ステップ 1. 前提条件の設定

Oracle TDE と の統合を実現するには AWS CloudHSM、以下が必要です。

- 少なくとも 1 つの HSM を持つアクティブな AWS CloudHSM クラスター。
- 次のソフトウェアがインストールされた Amazon Linux オペレーティングシステムを実行している Amazon EC2 インスタンス。 :
  - AWS CloudHSM クライアントおよびコマンドラインツール。
  - PKCS #11 用の AWS CloudHSM ソフトウェアライブラリ。
  - Oracle Database. は Oracle TDE 統合 AWS CloudHSM をサポートしています。クライアント SDK 5.6 以降は、Oracle Database 19c 用 Oracle TDE をサポートしています。クライアント SDK 3 は Oracle データベースバージョン 11g および 12c の Oracle TDE をサポートします。
- クラスター内の HSM で TDE マスター暗号化キーを所有および管理するための暗号化ユーザー (CU)。

これらの前提条件のすべてを設定するには、以下のステップを実行します。

との Oracle TDE 統合の前提条件を設定するには AWS CloudHSM

1. 「開始方法」のステップを完了します。これを行うと、1 つの HSM を含むアクティブなクラスターが提供されます。Amazon Linux オペレーティングシステムで実行される Amazon EC2 インスタンスも作成されます。 AWS CloudHSM クライアントツールとコマンドラインツールもインストールおよび設定されます。
2. (オプション) 他の HSM をクラスターに追加します。詳細については、「AWS CloudHSM クラスターへの HSM の追加」を参照してください。
3. Amazon EC2 クライアントインスタンスに接続し、以下を実行します。

   a. PKCS #11 用の AWS CloudHSM ソフトウェアライブラリをインストールします。

   b. Oracle Database をインストールします。詳細については、Oracle Database のドキュメントを参照してください。クライアント SDK 5.6 以降は、Oracle Database 19c 用 Oracle TDE をサポートしています。クライアント SDK 3 は Oracle データベースバージョン 11g および 12c の Oracle TDE をサポートします。

   c. cloudhsm_mgmt_util コマンドラインツールを使用し、クラスターで暗号化ユーザー (CU) を作成します。CU の作成に関する詳細については、CMU で HSM ユーザーを管理する方法と HSM ユーザー を参照してください。

ステップ 2: Oracle データベース設定を更新する

クラスターで HSM を外部セキュリティモジュールとして使用するように Oracle Database の設定を更新するには、次のステップを実行します。外部セキュリティモジュールの詳細については、Oracle Database Advanced Security ガイドの「透過的データ暗号化の概要」を参照してください。

Oracle 設定を更新するには

1. Amazon EC2 クライアントインスタンスに接続します。これは、Oracle Database をインストールした先のインスタンスです。
2. `sqlnet.ora` というファイルのバックアップコピーを作成します。このファイルの場所については、Oracle のドキュメントを参照してください。
3. テキストエディタを使用して、`sqlnet.ora` というファイルを編集します。次の行を追加します。ファイルの既存の行が `encryption_wallet_location` で始まる場合は、既存の行を次の行に置き換えます。

```
encryption_wallet_location=(source=(method=hsm))
```

ファイルを保存します。

4. 次のコマンドを実行して、Oracle Database が AWS CloudHSM PKCS #11 ソフトウェアライブラリのライブラリファイルを検索するディレクトリを作成します。

```
sudo mkdir -p /opt/oracle/extapi/64/hsm
```

5. 次のコマンドを実行して、PKCS #11 ファイルの AWS CloudHSM ソフトウェアライブラリを前のステップで作成したディレクトリにコピーします。

```
sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
```

> **Note**
>
> /opt/oracle/extapi/64/hsm ディレクトリに含めるライブラリファイルは 1 つに限られます。そのディレクトリに存在する他のファイルを削除します。

6. 次のコマンドを実行して、/opt/oracle ディレクトリおよびその内容すべての所有権を変更します。

```
sudo chown -R oracle:dba /opt/oracle
```

7. Oracle Database を起動します。

ステップ 3: Oracle TDE マスター暗号化キーの生成

クラスターの HSM で Oracle TDE マスターキーを生成するには、次の手順を実行します。

マスターキーを生成するには

1. 次のコマンドを使用して、Oracle SQL*Plus を開きます。プロンプトが表示されたら、Oracle Database のインストール時に設定したシステムパスワードを入力します。

```
sqlplus / as sysdba
```

> **Note**
>
> クライアント SDK 3 の場合、マスターキーを生成するたびに CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE 環境変数を設定する必要があります。この変数は、マスターキーの生成にのみ必要です。詳細については、サードパーティーアプリケーションの統合に関する既知の問題 の「問題：Oracleは CKA_MODIFIABLE マスターキーの生成中に PCKS #11 属性を設定しますが、HSM はそれをサポートしていません」を参照してください。

2. 次の例に示すように、マスター暗号化キーを作成する SQL ステートメントを実行します。使用しているバージョンの Oracle Database に対応するステートメントを使用します。*<CU user name>* を暗号化ユーザー (CU) のユーザー名に置き換えます。*<password>* を CU パスワードに置き換えます。

> **Important**
>
> 次のコマンドは 1 回のみ実行します。コマンドを実行するたびに、新しいマスター暗号化キーが作成されます。

- Oracle Database バージョン 11 の場合は、次の SQL ステートメントを実行します。

```
SQL> alter system set encryption key identified by "<CU user name>:<password>";
```

- Oracle Database バージョン 12 およびバージョン 19c の場合は、次の SQL ステートメントを実行します。

```
SQL> administer key management set key identified by "<CU user
 name>:<password>";
```

レスポンスが System altered または keystore altered の場合は、Oracle TDE のマスターキーが正常に生成および設定されています。

3. (オプション) 次のコマンドを実行して Oracle ウォレットのステータスを確認します。

```
SQL> select * from v$encryption_wallet;
```

ウォレットが開いていない場合は、次のいずれかのコマンドを使用して開きます。*<CU user name>* を暗号化ユーザー (CU) の名前に置き換えます。*<password>* を CU パスワードに置き換えます。

- Oracle 11 の場合は、次のコマンドを実行してウォレットを開きます。

```
SQL> alter system set encryption wallet open identified by "<CU user
 name>:<password>";
```

手動でウォレットを閉じるには、次のコマンドを実行します。

```
SQL> alter system set encryption wallet close identified by "<CU user
 name>:<password>";
```

- Oracle 12 および Oracle 19c の場合は、次のコマンドを実行してウォレットを開きます。

```
SQL> administer key management set keystore open identified by "<CU user
 name>:<password>";
```

手動でウォレットを閉じるには、次のコマンドを実行します。

```
SQL> administer key management set keystore close identified by "<CU user
 name>:<password>";
```

で Microsoft SignTool を使用してファイルに署名 AWS CloudHSM する

AWS CloudHSM は、Microsoft Signtool を使用してクライアント SDK 3 およびクライアント SDK 5 を介してファイルに署名するサポートを提供します。これらのツールを使用する手順は、現在ダウンロードしているクライアント SDK のバージョンによって異なります。次のセクションでは、各 SDK に関する情報を提供します。

トピック

- クライアント SDK 5 で Microsoft SignTool を使用してファイルに署名する
- クライアント SDK 3 で Microsoft SignTool を使用してファイルに署名する

クライアント SDK 5 で Microsoft SignTool を使用してファイルに署名する

暗号化やパブリックキー基盤 (PKI) では、デジタル署名は、データが信頼されたエンティティより送信されたことを確認することを目的として使用されます。署名は、データが送信中に改ざんされていないことも示します。署名とは、送信者のプライベートキーを使用して生成された暗号化ハッシュを指します。受信者は、ハッシュ署名を送信者のパブリックキーで復号することで、データの整合性を検証できます。また、送信者は、デジタル証明書を管理する責任があります。デジタル証明書は、送信者のプライベートキーの所有者を証明し、復号に必要なパブリックキーを受信者に渡します。プライベートキーが送信者によって所有されている限り、署名は信頼できます。 は、これらのキーを排他的なシングルテナントアクセスで保護するために、安全な FIPS 140-2 レベル 3 検証済みハードウェア AWS CloudHSM を提供します。

Microsoft SignTool は、ファイルに対して署名、検証、タイムスタンプ付与を行うコマンドラインツールであり、多くの組織で使用されています。 AWS CloudHSM を使用して、SignTool で必要になるまでキーペアを安全に保存できるため、データに署名するためのワークフローを簡単に自動化できます。

以下のトピックでは、 で SignTool を使用する方法の概要を説明します AWS CloudHSM。

トピック

- ステップ 1: 前提条件の設定
- ステップ 2: 署名用証明書を作成する
- ステップ 3: ファイルに署名する

ステップ 1: 前提条件の設定

で Microsoft SignTool を使用するには AWS CloudHSM、以下が必要です。

- Windows オペレーティングシステムが実行されている Amazon EC2 クライアントインスタンス。
- 認証局 (CA)。自己管理、またはサードパーティープロバイダーが作成したもの。
- EC2 インスタンスと同じ仮想パブリッククラウド (VPC) 内のアクティブな AWS CloudHSM クラスター。クラスターには、少なくとも 1 つの HSM が存在している必要があります。
- AWS CloudHSM クラスター内のキーを所有および管理するための Crypto User (CU)。
- 未署名のファイルまたは実行可能ファイル。
- Microsoft Windows Software Development Kit (SDK)。

Windows SignTool AWS CloudHSM で を使用するための前提条件を設定するには

1. このガイドの「開始方法」セクションの指示に従って、Windows EC2 インスタンスと AWS CloudHSM クラスターを起動します。
2. 独自の Windows Server CA をホストする場合は、「 で Windows Server を認証機関として設定 AWS CloudHSMする」のステップ 1 と 2 に従います。それ以外の場合は、公的に信頼されたサードパーティーの CA をそのまま使用します。
3. 次のバージョンの Microsoft Windows SDK を Windows EC2 インスタンスにダウンロードしてインストールします。

   - Microsoft Windows SDK 10
   - Microsoft Windows SDK 8.1
   - Microsoft Windows SDK 7

   `SignTool` 実行可能ファイルは、デスクトップアプリケーションのインストール機能用の Windows SDK Signing Tool に含まれます。不要な場合は、他の機能をインストール対象から除外することができます。デフォルトのインストール場所は次のとおりです。

   ```
   C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU
    architecture>\signtool.exe
   ```

Microsoft Windows SDK、 AWS CloudHSM クラスター、および CA を使用して署名証明書を作成できるようになりました。

ステップ 2: 署名用証明書を作成する

EC2 インスタンスに Windows SDK をダウンロードしたら、これを使用して証明書署名リクエスト (CSR) を生成することができます。CSR は、未署名の証明書であり、署名用に最終的に CA に渡されます。この例では、Windows SDK に含まれる`certreq` 実行可能ファイルを使用して、CSR を生成します。

`certreq` 実行可能ファイルを使用して CSR を生成するには

1. Windows EC2 インスタンスに接続されていない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。

2. 以下の行に含まれる `request.inf` ファイルを作成します。`Subject` 情報をお客様の組織の情報に置き換えます。各パラメータの説明については、「Microsoft のドキュメント」を参照してください。

```
[Version]
Signature= $Windows NT$
[NewRequest]
Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-Unit>,L=<City>,S=<State>"
RequestType=PKCS10
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "CloudHSM Key Storage Provider"
KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
MachineKeySet = True
Exportable = False
```

3. `certreq.exe` を実行します。この例では、CSR を `request.csr` として保存します。

```
certreq.exe -new request.inf request.csr
```

内部的には、 AWS CloudHSM クラスターに新しいキーペアが生成され、そのペアのプライベートキーを使用して CSR が作成されます。

4. CA に CSR を送ります。Windows Server CA を使用している場合は、次のステップを行います。

   a. 次のコマンドを入力して、CA ツールを開きます。

   ```
   certsrv.msc
   ```

   b. 新しいウィンドウで、CA サーバーの名前を右クリックします。[すべてのタスク]、[Submit new request (新しいリクエストの送信)] の順に選択します。

   c. `request.csr` の場所に移動し、[開く] を選択します。

   d. サーバー CA メニューから、[保留中のリクエスト] フォルダを表示します。先ほど作成したリクエストを右クリックし、[すべてのタスク] で [問題] を選択します。

   e. [Issued Certificates (発行済みの証明書)] フォルダ ([保留中のリクエスト] フォルダの上) に移動します。

   f. [開く] を選択して証明書を表示し、[詳細] タブを選択します。

g. [Copy to File (ファイルにコピー)] を選択して、証明書のエクスポートウィザードを起動します。DER でエンコードされた X.509 ファイルを `signedCertificate.cer` として安全な場所に保存します。

h. CA ツールを終了し、次のコマンドを使用して、証明書ファイルを Windows の Personal Certificate Store に移動します。他のアプリケーションで使用できます。

```
certreq.exe -accept signedCertificate.cer
```

これで、インポートしたファイルを使用して、ファイルに署名する ことができます。

ステップ 3: ファイルに署名する

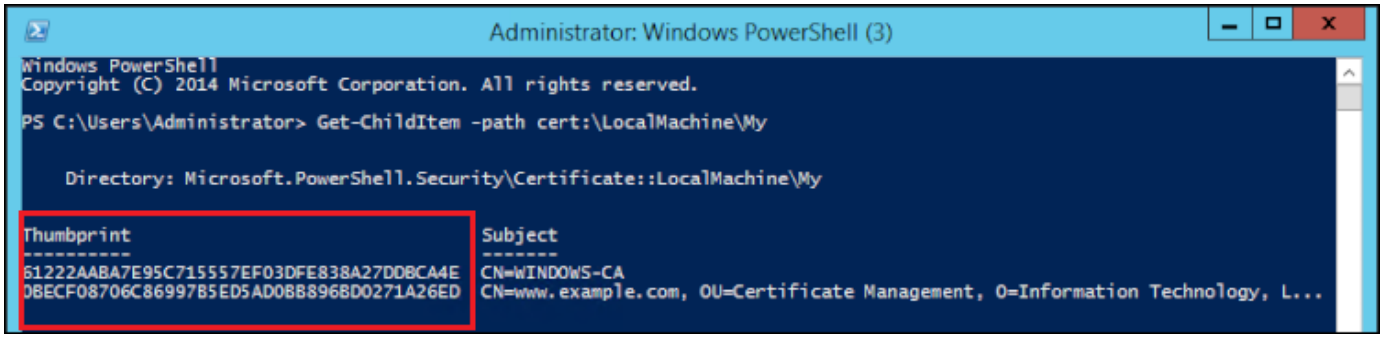
これで、SignTool と、インポートした証明書を使用して、サンプルファイルに署名することができます。そのためには、証明書の SHA-1 ハッシュ、またはサムプリントを把握しておく必要があります。サムプリントを使用することで、 AWS CloudHSMによって検証された証明書のみ SignTool で使用されるようできます。この例では、PowerShell を使用して証明書のハッシュを取得します。また、CA の GUI または Windows SDK の `certutil` 実行可能ファイルを使用することもできます。

証明書のサムプリントを取得し、それを使用してファイルに署名するには

1. 管理者として PowerShell を開き、次のコマンドを実行します。

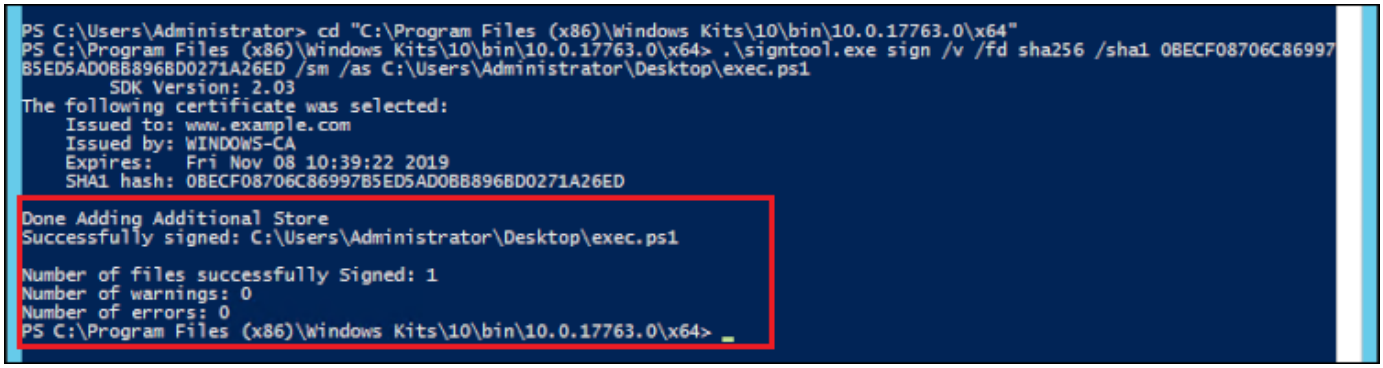
```
Get-ChildItem -path cert:\LocalMachine\My
```

返った `Thumbprint` をコピーします。



2. `SignTool.exe` がある PowerShell 内のディレクトリに移動します。デフォルトの場所は `C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64` です。
3. 最後に、次のコマンドを実行してファイルに署名します。コマンドが正常に実行されると、PowerShell より成功のメッセージが返ります。

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator
\Desktop\<test>.ps1
```



4. (オプション) ファイルの署名を検証するには、次のコマンドを使用します。

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

クライアント SDK 3 で Microsoft SignTool を使用してファイルに署名する

暗号化やパブリックキー基盤 (PKI) では、デジタル署名は、データが信頼されたエンティティより送信されたことを確認することを目的として使用されます。署名は、データが送信中に改ざんされていないことも示します。署名とは、送信者のプライベートキーを使用して生成された暗号化ハッシュを指します。受信者は、ハッシュ署名を送信者のパブリックキーで復号することで、データの整合性を検証できます。また、送信者は、デジタル証明書を管理する責任があります。デジタル証明書は、送信者のプライベートキーの所有者を証明し、復号に必要なパブリックキーを受信者に渡します。プライベートキーが送信者によって所有されている限り、署名は信頼できます。 は、これらのキーを排他的なシングルテナントアクセスで保護するために、安全な FIPS 140-2 レベル 3 検証済みハードウェア AWS CloudHSM を提供します。

Microsoft SignTool は、ファイルに対して署名、検証、タイムスタンプ付与を行うコマンドラインツールであり、多くの組織で使用されています。 AWS CloudHSM を使用して、SignTool で必要になるまでキーペアを安全に保存できるため、データに署名するためのワークフローを簡単に自動化できます。

以下のトピックでは、 で SignTool を使用する方法の概要を説明します AWS CloudHSM。

トピック

- ステップ 1: 前提条件の設定

- ステップ 2: 署名用証明書を作成する
- ステップ 3: ファイルに署名する

ステップ 1: 前提条件の設定

で Microsoft SignTool を使用するには AWS CloudHSM、以下が必要です。

- Windows オペレーティングシステムが実行されている Amazon EC2 クライアントインスタンス。
- 認証局 (CA)。自己管理、またはサードパーティープロバイダーが作成したもの。
- EC2 インスタンスと同じ仮想パブリッククラウド (VPC) 内のアクティブな AWS CloudHSM クラスター。クラスターには、少なくとも 1 つの HSM が存在している必要があります。
- AWS CloudHSM クラスター内のキーを所有および管理するための Crypto User (CU)。
- 未署名のファイルまたは実行可能ファイル。
- Microsoft Windows Software Development Kit (SDK)。

Windows SignTool AWS CloudHSM で を使用するための前提条件を設定するには

1. このガイドの「開始方法」セクションの指示に従って、Windows EC2 インスタンスと AWS CloudHSM クラスターを起動します。
2. 独自の Windows Server CA をホストする場合は、「 で Windows Server を認証機関として設定 AWS CloudHSMする」のステップ 1 と 2 に従います。それ以外の場合は、公的に信頼されたサードパーティーの CA をそのまま使用します。
3. 次のバージョンの Microsoft Windows SDK を Windows EC2 インスタンスにダウンロードしてインストールします。

   - Microsoft Windows SDK 10
   - Microsoft Windows SDK 8.1
   - Microsoft Windows SDK 7

   SignTool 実行可能ファイルは、デスクトップアプリケーションのインストール機能用の Windows SDK Signing Tool に含まれます。不要な場合は、他の機能をインストール対象から除外することができます。デフォルトのインストール場所は次のとおりです。

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU
 architecture>\signtool.exe
```

Microsoft Windows SDK、 AWS CloudHSM クラスター、および CA を使用して署名証明書を作成できるようになりました。

ステップ 2: 署名用証明書を作成する

EC2 インスタンスに Windows SDK をダウンロードしたら、これを使用して証明書署名リクエスト (CSR) を生成することができます。CSR は、未署名の証明書であり、署名用に最終的に CA に渡されます。この例では、Windows SDK に含まれる`certreq` 実行可能ファイルを使用して、CSR を生成します。

`certreq` 実行可能ファイルを使用して CSR を生成するには

1. Windows EC2 インスタンスに接続されていない場合は、接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. 以下の行に含まれる `request.inf` ファイルを作成します。`Subject` 情報をお客様の組織の情報に置き換えます。各パラメータの説明については、「Microsoft のドキュメント」を参照してください。

```
[Version]
Signature= $Windows NT$
[NewRequest]
Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-
Unit>,L=<City>,S=<State>"
RequestType=PKCS10
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "Cavium Key Storage Provider"
KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
MachineKeySet = True
Exportable = False
```

3. `certreq.exe` を実行します。この例では、CSR を `request.csr` として保存します。

```
certreq.exe -new request.inf request.csr
```

内部的には、 AWS CloudHSM クラスターに新しいキーペアが生成され、そのペアのプライベートキーを使用して CSR が作成されます。

4. CA に CSR を送ります。Windows Server CA を使用している場合は、次のステップを行います。

a. 次のコマンドを入力して、CA ツールを開きます。

```
certsrv.msc
```

b. 新しいウィンドウで、CA サーバーの名前を右クリックします。[すべてのタスク]、[Submit new request (新しいリクエストの送信)] の順に選択します。

c. `request.csr` の場所に移動し、[開く] を選択します。

d. サーバー CA メニューから、[保留中のリクエスト] フォルダを表示します。先ほど作成したリクエストを右クリックし、[すべてのタスク] で [問題] を選択します。

e. [Issued Certificates (発行済みの証明書)] フォルダ ([保留中のリクエスト] フォルダの上) に移動します。

f. [開く] を選択して証明書を表示し、[詳細] タブを選択します。

g. [Copy to File (ファイルにコピー)] を選択して、証明書のエクスポートウィザードを起動します。DER でエンコードされた X.509 ファイルを `signedCertificate.cer` として安全な場所に保存します。

h. CA ツールを終了し、次のコマンドを使用して、証明書ファイルを Windows の Personal Certificate Store に移動します。他のアプリケーションで使用できます。

```
certreq.exe -accept signedCertificate.cer
```

これで、インポートしたファイルを使用して、ファイルに署名する ことができます。

ステップ 3: ファイルに署名する

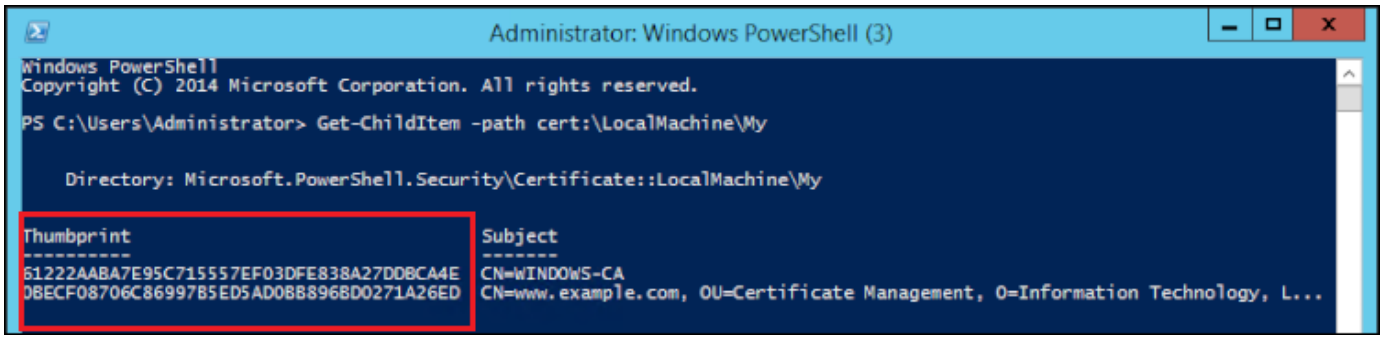
これで、SignTool と、インポートした証明書を使用して、サンプルファイルに署名することができます。そのためには、証明書の SHA-1 ハッシュ、またはサムプリントを把握しておく必要があります。サムプリントを使用することで、 AWS CloudHSMによって検証された証明書のみ SignTool で使用されるようできます。この例では、PowerShell を使用して証明書のハッシュを取得します。また、CA の GUI または Windows SDK の `certutil` 実行可能ファイルを使用することもできます。

証明書のサムプリントを取得し、それを使用してファイルに署名するには

1. 管理者として PowerShell を開き、次のコマンドを実行します。

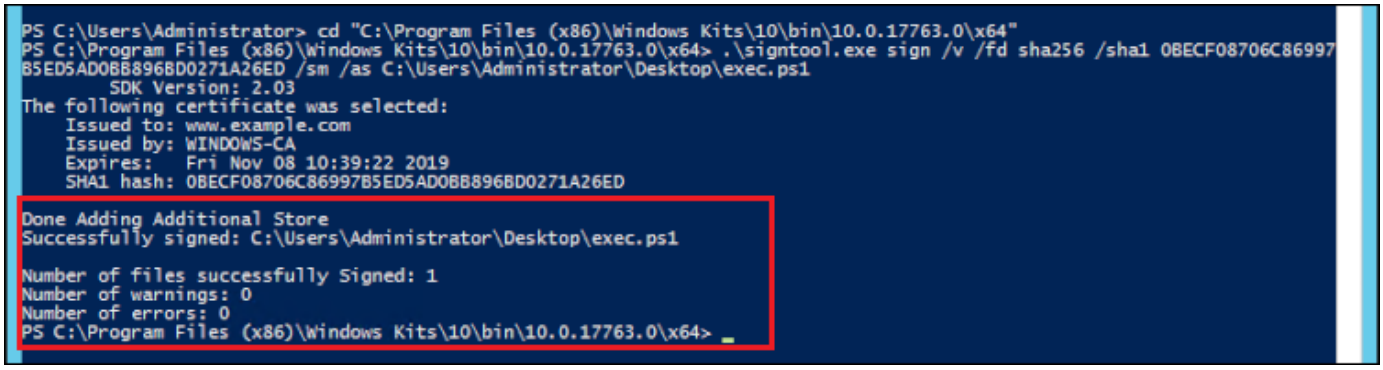
```
Get-ChildItem -path cert:\LocalMachine\My
```

返った Thumbprint をコピーします。



2. SignTool.exe がある PowerShell 内のディレクトリに移動します。デフォルトの場所は C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64 です。
3. 最後に、次のコマンドを実行してファイルに署名します。コマンドが正常に実行されると、PowerShell より成功のメッセージが返ります。

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator
\Desktop\<test>.ps1
```



4. (オプション) ファイルの署名を検証するには、次のコマンドを使用します。

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Java Keytool と Jarsigner の統合 AWS CloudHSM

AWS CloudHSM は、クライアント SDK 3 およびクライアント SDK 5 を介して Java Keytool および Jarsigner ユーティリティとの統合を提供します。これらのツールを使用する手順は、現在ダウンロードしているクライアント SDK のバージョンによって異なります。次のセクションでは、各 SDK に関する情報を提供します。

トピック

- クライアント SDK 5 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合する
- Client SDK 3 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合する

クライアント SDK 5 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合する

AWS CloudHSM key store は、ハードウェアセキュリティモジュール (HSM) のキーに関連付けられた証明書を、 `keytool`や などのサードパーティーツールを使用して使用する専用 JCE キーストアです`jarsigner`。証明書はパブリックで非機密データであるため、HSM に証明書を保存 AWS CloudHSM しません。 AWS CloudHSM キーストアは証明書をローカルファイルに格納し、証明書を HSM の対応するキーにマッピングします。

AWS CloudHSM キーストアを使用して新しいキーを生成すると、ローカルキーストアファイルにエントリは生成されず、キーは HSM に作成されます。同様に、 AWS CloudHSM キーストアを使用してキーを検索すると、検索が HSM に渡されます。証明書を AWS CloudHSM キーストアに保存すると、プロバイダーは対応するエイリアスを持つキーペアが HSM に存在することを確認し、提供された証明書を対応するキーペアに関連付けます。

トピック

- クライアント SDK 5 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合するための前提条件
- クライアント SDK 5 を使用して AWS CloudHSM keytool でキーストアを使用する
- クライアント SDK 5 を使用して Jarsigner で AWS CloudHSM キーストアを使用する
- クライアント SDK 5 を使用した Java Keytool と Jarsigner AWS CloudHSM の統合に関する既知の問題

クライアント SDK 5 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合するための前提条件

AWS CloudHSM キーストアを使用するには、まず AWS CloudHSM JCE SDK を初期化して設定する必要があります。これを行うには、次の手順を実行します。

手順 1: JCE をインストールする

AWS CloudHSM クライアントの前提条件を含む JCE をインストールするには、Java ライブラリをインストールするステップに従います。

手順 2: 環境変数に HSM ログイン認証情報を追加する

HSM ログイン認証情報を格納する環境変数を設定します。

Linux

```
$ export HSM_USER=<HSM user name>
```

```
$ export HSM_PASSWORD=<HSM password>
```

Windows

```
PS C:\> $Env:HSM_USER=<HSM user name>
```

```
PS C:\> $Env:HSM_PASSWORD=<HSM password>
```

> **Note**
>
> AWS CloudHSM JCE にはさまざまなログインオプションがあります。サードパーティーアプリケーションで AWS CloudHSM キーストアを使用するには、環境変数で暗黙的なログインを使用する必要があります。アプリケーションコードによる明示的なログインを使用する場合は、 AWS CloudHSM キーストアを使用して独自のアプリケーションを構築する必要があります。詳細については、AWS CloudHSM 「 キーストアの使用」の記事を参照してください。

手順 3: JCE プロバイダーを登録する

Java クラウドプロバイダーの設定で JCE プロバイダーを登録するには、以下の手順を実行します。

1. Java インストールで `java.security` 設定ファイルを開き、編集します。
2. `java.security` 設定ファイル
   で、`com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider` を最後のプロバイダー

として追加します。例えば、java.security ファイルに 9 つのプロバイダーがある場合は、セクションの最後のプロバイダーとして次のプロバイダーを追加します。

```
security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
```

> **Note**
>
> AWS CloudHSM プロバイダーをより高い優先度として追加すると、ソフトウェアに安全にオフロードされる可能性のあるオペレーションに AWS CloudHSM プロバイダーが優先されるため、システムのパフォーマンスに悪影響を及ぼす可能性があります。ベストプラクティスとして、 AWS CloudHSM またはソフトウェアベースのプロバイダーに関係なく、オペレーションに使用するプロバイダーを常に指定します。

> **Note**
>
> AWS CloudHSM キーストアで keytool を使用してキーを生成するときに -providerName、-providerclass、-providerpath コマンドラインオプションを指定すると、エラーが発生する可能性があります。

クライアント SDK 5 を使用して AWS CloudHSM keytool でキーストアを使用する

keytool は、一般的なキーおよび証明書タスク向けの一般的なコマンドラインユーティリティです。keytool に関する完全なチュートリアルは、 AWS CloudHSM ドキュメントの範囲外です。この記事では、 をキーストアを通じて信頼のルート AWS CloudHSM として使用するときに、さまざまな AWS CloudHSM keytool 関数で使用する特定のパラメータについて説明します。

keytool をキー AWS CloudHSM ストアで使用する場合は、任意の keytool コマンドに次の引数を指定します。

> **ML-DSA と keytool**
>
> Keytool は 24 以降の ML-DSA JDK をサポートしています。
> ML-DSA キーでは、次の keytool コマンドがサポートされています。
>
> - -certreq – ML-DSA キーの証明書署名リクエストを生成します。
> - -importcert – ML-DSA キーの証明書をインポートします。

- -exportcert – ML-DSA キーの証明書をエクスポートします。

次のコマンドは、ML-DSA キーではサポートされていません。

- -genkeypair – AWS CloudHSM は ML-DSA キーのインポートをサポートしておらず、インポート前にソフトウェアでキーkeytoolを生成します。KeyPairGenerator または CloudHSM CLI を使用して、ML-DSA キーペアを生成します。

Linux

```
-storetype CLOUDHSM -J-classpath< '-J/opt/cloudhsm/java/*'>
```

Windows

```
-storetype CLOUDHSM -J-classpath<'-J"C:\Program Files\Amazon\CloudHSM\java\*"'>
```

キーストアを使用して新しい AWS CloudHSM キーストアファイルを作成する場合は、「」を参照してくださいAWS CloudHSM クライアント SDK 3 に AWS CloudHSM KeyStoreを使用する。既存のキーストアを使用するには、keytool の –keystore 引数を使用して、その名前 (パスを含む) を指定します。keytool コマンドで存在しないキーストアファイルを指定すると、 AWS CloudHSM キーストアは新しいキーストアファイルを作成します。

keytool を使用して新しい AWS CloudHSM キーを作成する

keytool を使用して、 AWS CloudHSM JCE SDK でサポートされているキーの RSA、EC (Ed25519)、AES、および DESede タイプを生成できます。

Important

keytool で生成されたキーはソフトウェアで生成され、抽出可能な永続キー AWS CloudHSM として にインポートされます。

ML-DSA キー生成

AWS CloudHSM は、 による ML-DSA keytool キーの生成をサポートしていません-genkeypair。KeyPairGenerator または CloudHSM CLI を使用して ML-DSA キーペアを生

成し、 を使用して CloudHSM KeyStore に保存しますKeyStore.setKeyEntry()。例については、 GitHubウェブサイトのMldsaKeyStoreExampleRunner」を参照してください。

エクスポートできないキーを keytool の外部で生成したうえで、対応する証明書をキーストアにインポートすることを強くお勧めします。keytool と Jarsigner を介して抽出可能な RSA キーまたは EC キーを使用する場合、プロバイダーは からキーをエクスポートし AWS CloudHSM 、ローカルでキーを使用してオペレーションに署名します。

複数のクライアントインスタンスが AWS CloudHSM クラスターに接続されている場合、1 つのクライアントインスタンスのキーストアに証明書をインポートしても、他のクライアントインスタンスで証明書が自動的に使用可能になるわけではないことに注意してください。各クライアントインスタンスでキーおよび関連する証明書を登録するには、「the section called “keytool を使用して CSR を生成する”」の説明に従って Java アプリケーションを実行する必要があります。または、1 つのクライアントで必要な変更を行い、結果のキーストアファイルを他のすべてのクライアントインスタンスにコピーすることもできます。

例 1: 対称型 AES-256 キーを生成し、作業ディレクトリの「example_keystore.store」という名前のキーストアファイルに保存します。*<secret label>* を独自のラベルに置き換えます。

Linux

```
$ keytool -genseckey -alias <secret label> -keyalg aes \
 -keysize 256 -keystore example_keystore.store \
 -storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
```

Windows

```
PS C:\> keytool -genseckey -alias <secret label> -keyalg aes `
 -keysize 256 -keystore example_keystore.store `
 -storetype CloudHSM -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

例 2: RSA 2,048 キーペアを生成し、作業ディレクトリの「example_keystore.store」という名前のキーストアファイルに保存します。*<RSA key pair label>* を独自のラベルに置き換えます。

Linux

```
$ keytool -genkeypair -alias <RSA key pair label> \
 -keyalg rsa -keysize 2048 \
```

```
 -sigalg sha512withrsa \
 -keystore example_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -genkeypair -alias <RSA key pair label> `
 -keyalg rsa -keysize 2048 `
 -sigalg sha512withrsa `
 -keystore example_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

サポートされている署名アルゴリズムのリストは、Java ライブラリにあります。

keytool を使用して AWS CloudHSM キーを削除する

AWS CloudHSM キーストアはキーの削除をサポートしていません。Destroyable インターフェイスの destroy メソッドを使用してキーを削除できます。

```
((Destroyable) key).destroy();
```

keytool AWS CloudHSM を使用して CSR を生成する

AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine を使用すると、証明書署名要求 (CSR) を柔軟に生成できます。次のコマンドは、keytool を使用して、エイリアス `example-key-pair` を持つキーペアの CSR を生成します。

Linux

```
$ keytool -certreq -alias <key pair label> \
 -file my_csr.csr \
 -keystore example_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -certreq -alias <key pair label> `
```

```
 -file my_csr.csr `
 -keystore example_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

> **Note**
>
> keytool のキーペアを使用するには、指定されたキーストアファイルにそのキーペアのエントリが必要です。keytool の外部で生成されたキーペアを使用する場合は、キーおよび証明書のメタデータをキーストアにインポートする必要があります。キーストアデータをインポートする手順については、「the section called “keytool を使用して証明書をキー ストアにインポートする ”」を参照してください。

keytool を使用して中間証明書とルート証明書を AWS CloudHSM キーストアにインポートする

CA 証明書を にインポートするには AWS CloudHSM、新しくインポートされた証明書で完全な証明書チェーンの検証を有効にする必要があります。次のコマンドでは、例を示しています。

Linux

```
$ keytool -import -trustcacerts -alias rootCAcert \
 -file rootCAcert.cert -keystore example_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -import -trustcacerts -alias rootCAcert `
 -file rootCAcert.cert -keystore example_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

複数のクライアントインスタンスを AWS CloudHSM クラスターに接続する場合、1 つのクライアントインスタンスのキーストアに証明書をインポートしても、他のクライアントインスタンスで証明書が自動的に使用可能になることはありません。各クライアントインスタンスで証明書をインポートする必要があります。

keytool を使用して AWS CloudHSM キーストアから証明書を削除する

次のコマンドは、Java keytool キーストアから AWS CloudHSM 証明書を削除する方法の例を示しています。

Linux

```
$ keytool -delete -alias mydomain \
 -keystore example_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -delete -alias mydomain `
 -keystore example_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

複数のクライアントインスタンスを AWS CloudHSM クラスターに接続しても、1 つのクライアントインスタンスのキーストアで証明書を削除しても、他のクライアントインスタンスから証明書は自動的に削除されません。各クライアントインスタンスで証明書を削除する必要があります。

keytool を使用して作業証明書を AWS CloudHSM キーストアにインポートする

証明書署名要求 (CSR) が署名されると、それを AWS CloudHSM キーストアにインポートし、適切なキーペアに関連付けることができます。次のコマンドでは、例を示しています。

Linux

```
$ keytool -importcert -noprompt -alias <key pair label> \
 -file my_certificate.crt \
 -keystore example_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -importcert -noprompt -alias <key pair label> `
 -file my_certificate.crt `
```

```
 -keystore example_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

エイリアスは、キーストア内の関連付けられた証明書を持つキーペアである必要があります。キーが keytool の外部で生成される場合や、別のクライアントインスタンスで生成される場合は、まずキーおよび証明書のメタデータをキーストアにインポートする必要があります。

証明書チェーンは検証可能である必要があります。証明書を検証できない場合は、チェーンを検証できるように、署名 (証明機関) 証明書をキーストアにインポートする必要があります。

keytool AWS CloudHSM を使用して から証明書をエクスポートする

次の例では、バイナリ X.509 形式の証明書を生成します。人間が読み取り可能な証明書をエクスポートするには AWS CloudHSM、 `-exportcert` コマンド`-rfc`に を追加します。

Linux

```
$ keytool -exportcert -alias <key pair label> \
 -file my_exported_certificate.crt \
 -keystore example_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -exportcert -alias <key pair label> `
 -file my_exported_certificate.crt `
 -keystore example_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

クライアント SDK 5 を使用して Jarsigner で AWS CloudHSM キーストアを使用する

Jarsigner は、ハードウェア セキュリティ モジュール (HSM) に安全に保存されたキーを使用して JAR ファイルに署名するための一般的なコマンドラインユーティリティです。Jarsigner に関する完全なチュートリアルは、 AWS CloudHSM ドキュメントの対象外です。このセクションでは、AWS CloudHSM キーストアを通じて信頼のルート AWS CloudHSM として署名と検証に使用する Jarsigner パラメータについて説明します。

ML-DSA jarsigner の要件

ML-DSA JDK 署名アルゴリズム (など`-sigalg ML-DSA-44`) で使用する場合、26 以降が必要ですjarsigner。

Jarsigner で AWS CloudHSM キーと証明書をセットアップする

Jarsigner で AWS CloudHSM JAR ファイルに署名する前に、次の手順を設定または完了していることを確認してください。

1. 「AWS CloudHSM キーストアの前提条件」のガイダンスに従います。
2. 現在のサーバーまたはクライアントインスタンスの AWS CloudHSM キーストアに保存する必要がある署名キー、関連する証明書、証明書チェーンを設定します。でキーを作成し AWS CloudHSM 、関連するメタデータを AWS CloudHSM キーストアにインポートします。keytool を使用してキーおよび証明書を設定する場合は、「the section called “keytool で新しい キーを作成する”」を参照してください。複数のクライアントインスタンスを使用して JAR に署名する場合は、キーを作成し、証明書チェーンをインポートします。次に、結果のキーストアファイルを各クライアントインスタンスにコピーします。新しいキーを頻繁に生成する場合は、各クライアントインスタンスに証明書を個別にインポートする方が簡単です。
3. 証明書チェーン全体が検証可能である必要があります。証明書チェーンを検証できるようにするには、CA 証明書と中間証明書を AWS CloudHSM キーストアに追加する必要がある場合があります。Java コードを使用して証明書チェーンを検証する方法については、「the section called “JAR ファイルに署名する”」のコードスニペットを参照してください。必要に応じて、keytool を使用して証明書をインポートできます。keytool を使用する手順については、「the section called “keytool を使用して証明書をキー ストアにインポートする ”」を参照してください。

AWS CloudHSM と Jarsigner を使用して JAR ファイルに署名する

RSA キーまたは ECDSA キーによる署名と検証

次のコマンドを使用して、 AWS CloudHSM と で RSA または ECDSA キーを使用して JAR ファイルに署名しますjarsigner。

Linux;

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -keystore example_keystore.store \
```

```
 -signedjar signthisclass_signed.jar \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass.jar <key pair label>
```

Windows

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -keystore example_keystore.store `
 -signedjar signthisclass_signed.jar `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
  "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass.jar <key pair label>
```

署名付き JAR を確認するには、次のコマンドを使用します。

Linux

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -verify \
 -keystore example_keystore.store \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass_signed.jar <key pair label>
```

Windows

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -verify `
```

```
 -keystore example_keystore.store `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
 "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass_signed.jar <key pair label>
```

ML-DSA キーによる署名と検証

次のコマンドを使用して、 AWS CloudHSM と で ML-DSA キーを使用して JAR ファイルに署名しますjarsigner。

Linux

26 OpenJDK 以降の場合

```
jarsigner -keystore example_keystore.store \
 -signedjar signthisclass_signed.jar \
 -sigalg ML-DSA-44 \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass.jar <key pair label>
```

Windows

26 OpenJDK 以降の場合

```
jarsigner -keystore example_keystore.store `
 -signedjar signthisclass_signed.jar `
 -sigalg ML-DSA-44 `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
  "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass.jar <key pair label>
```

次のコマンドを使用して、ML-DSA キーで署名された JAR を検証します。

Linux

26 OpenJDK 以降の場合

```
jarsigner -verify \
 -keystore example_keystore.store \
 -sigalg ML-DSA-44 \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass_signed.jar <key pair label>
```

Windows

26 OpenJDK 以降の場合

```
jarsigner -verify `
 -keystore example_keystore.store `
 -sigalg ML-DSA-44 `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
 "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass_signed.jar <key pair label>
```

Ed25519 キーを使用した署名と検証

次のコマンドを使用して、 AWS CloudHSM と で Ed25519 キーを使用して JAR ファイルに署名しますjarsigner。

Linux

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -keystore example_keystore.store \
 -signedjar signthisclass_signed.jar \
 -sigalg Ed25519 \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass.jar <key pair label>
```

Windows

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -keystore example_keystore.store `
 -signedjar signthisclass_signed.jar `
 -sigalg Ed25519 `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
  "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass.jar <key pair label>
```

次のコマンドを使用して、Ed25519 キーで署名された JAR を検証します。

Linux

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -verify \
 -keystore example_keystore.store \
 -sigalg Ed25519 \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass_signed.jar <key pair label>
```

Windows

OpenJDK 17、OpenJDK 21、および OpenJDK 25 の場合

```
jarsigner -verify `
 -keystore example_keystore.store `
 -sigalg Ed25519 `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
 "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass_signed.jar <key pair label>
```

クライアント SDK 5 を使用した Java Keytool と Jarsigner AWS CloudHSM の統合に関する既知の問題

次のリストは、 AWS CloudHSM と、クライアント SDK 5 を使用した Java Keytool と Jarsigner との統合に関する既知の問題の現在のリストを示しています。

1. Keytool および Jarsigner ではnon-Ed25519 EC キーはサポートされていません。
2. keytool による ML-DSA キー生成はサポートされていません。`KeyPairGenerator` または CloudHSM CLI を使用して、ML-DSA キーペアを生成します。
3. を使用した ML-DSA 署名アルゴリズムには 26 JDK 以降jarsignerが必要です。

Client SDK 3 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合する

AWS CloudHSM key store は、ハードウェアセキュリティモジュール (HSM) のキーに関連付けられた証明書を、 `keytool`や などのサードパーティーツールを使用して使用する専用 JCE キーストアです`jarsigner`。証明書はパブリックで非機密データであるため、HSM に証明書を保存 AWS CloudHSM しません。 AWS CloudHSM キーストアは証明書をローカルファイルに格納し、証明書を HSM の対応するキーにマッピングします。

AWS CloudHSM キーストアを使用して新しいキーを生成すると、ローカルキーストアファイルにエントリは生成されず、キーは HSM に作成されます。同様に、 AWS CloudHSM キーストアを使用してキーを検索すると、検索が HSM に渡されます。証明書を AWS CloudHSM キーストアに保存すると、プロバイダーは対応するエイリアスを持つキーペアが HSM に存在することを確認し、提供された証明書を対応するキーペアに関連付けます。

トピック

- クライアント SDK 3 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合するための前提条件
- クライアント SDK 3 を使用して AWS CloudHSM keytool でキーストアを使用する
- クライアント SDK 3 を使用して Jarsigner で AWS CloudHSM キーストアを使用する
- クライアント SDK 3 を使用した Java Keytool と Jarsigner AWS CloudHSM の統合に関する既知の問題

- 既存のキーを AWS CloudHSM キーストアに登録する

クライアント SDK 3 を使用して Java Keytool および Jarsigner AWS CloudHSM と統合するための前提条件

AWS CloudHSM キーストアを使用するには、まず AWS CloudHSM JCE SDK を初期化して設定する必要があります。これを行うには、次の手順を実行します。

手順 1: JCE をインストールする

AWS CloudHSM クライアントの前提条件を含む JCE をインストールするには、Java ライブラリをインストールするステップに従います。

手順 2: 環境変数に HSM ログイン認証情報を追加する

HSM ログイン認証情報を格納する環境変数を設定します。

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

> **Note**
>
> CloudHSM JCE には、さまざまなログインオプションがあります。サードパーティーアプリケーションで AWS CloudHSM キーストアを使用するには、環境変数で暗黙的なログインを使用する必要があります。アプリケーションコードによる明示的なログインを使用する場合は、 AWS CloudHSM キーストアを使用して独自のアプリケーションを構築する必要があります。詳細については、AWS CloudHSM 「 キーストアの使用」の記事を参照してください。

手順 3: JCE プロバイダーを登録する

JCE プロバイダーを登録するには、Java クラウドプロバイダーの設定を使用します。

1. Java インストールで java.security 設定ファイルを開き、編集します。
2. java.security 設定ファイルで、`com.cavium.provider.CaviumProvider` を最後のプロバイダーとして追加します。たとえば、java.security ファイルに 9 つのプロバイダーがある場合は、

セクションの最後のプロバイダーとして次のプロバイダーを追加します。Cavium プロバイダーの優先順位を高く設定すると、システムのパフォーマンスに悪影響を与える可能性があります。

```
security.provider.10=com.cavium.provider.CaviumProvider
```

> **Note**
>
> パワーユーザーは、keytool を使用する際、セキュリティ構成ファイルを更新する代わりに `-providerName`、`-providerclass` および `-providerpath` のコマンドラインオプションを指定することに慣れているかもしれませんが、 AWS CloudHSM キーストアでキーを生成するときにコマンドラインオプションを指定しようとすると、エラーが発生します。

クライアント SDK 3 を使用して AWS CloudHSM keytool でキーストアを使用する

keytool は、Linux システム上の一般的なキーおよび証明書タスク向けの一般的なコマンドラインユーティリティです。keytool に関する完全なチュートリアルは、 AWS CloudHSM ドキュメントの範囲外です。この記事では、 をキーストアを通じて信頼のルート AWS CloudHSM として使用する場合に、さまざまな AWS CloudHSM keytool 関数で使用する特定のパラメータについて説明します。

keytool をキー AWS CloudHSM ストアで使用する場合は、任意の keytool コマンドに次の引数を指定します。

```
-storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib
```

キーストアを使用して新しい AWS CloudHSM キーストアファイルを作成する場合は、「」を参照してくださいAWS CloudHSM クライアント SDK 3 に AWS CloudHSM KeyStoreを使用する。既存のキーストアを使用するには、keytool の –keystore 引数を使用して、その名前 (パスを含む) を指定します。keytool コマンドで存在しないキーストアファイルを指定すると、 AWS CloudHSM キーストアは新しいキーストアファイルを作成します。

keytool を使用して新しい AWS CloudHSM キーを作成する

keytool を使用して、 AWS CloudHSM JCE SDK でサポートされている任意のタイプのキーを生成できます。キーと長さの完全なリストについては、Java ライブラリの「サポートされるキー」の記事を参照してください。

> **Important**
>
> keytool で生成されたキーはソフトウェアで生成され、抽出可能な永続キー AWS CloudHSM として にインポートされます。

抽出不可能なキーをハードウェアセキュリティモジュール (HSM) で直接作成し、keytool または Jarsigner で使用する手順は、「既存のキーを AWS CloudHSM キーストアに登録する」のコードサンプルに示されています。エクスポートできないキーを keytool の外部で生成したうえで、対応する証明書をキーストアにインポートすることを強くお勧めします。keytool と jarsigner を使用して抽出可能な RSA キーまたは EC キーを使用する場合、プロバイダーは からキーをエクスポート AWS CloudHSM し、ローカルでキーを使用して署名操作を行います。

CloudHSM クラスターに複数のクライアントインスタンスが接続されている場合、1 つのクライアントインスタンスのキーストアに証明書をインポートしても、他のクライアントインスタンスで自動的に使用できるようにはなりません。各クライアントインスタンスでキーおよび関連する証明書を登録するには、「keytool を使用して CSR を生成する」の説明に従って Java アプリケーションを実行する必要があります。または、1 つのクライアントで必要な変更を行い、結果のキーストアファイルを他のすべてのクライアントインスタンスにコピーすることもできます。

例 1: 対称型 AES-256 キーを生成し、作業ディレクトリの「example_keystore.store」という名前のキーストアファイルに保存します。*<secret label>* を独自のラベルに置き換えます。

```
keytool -genseckey -alias <secret label> -keyalg aes \
  -keysize 256 -keystore example_keystore.store \
  -storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib/
```

例 2: RSA 2,048 キーペアを生成し、作業ディレクトリの「example_keystore.store」という名前のキーストアファイルに保存します。*<RSA key pair label>* を独自のラベルに置き換えます。

```
keytool -genkeypair -alias <RSA key pair label> \
        -keyalg rsa -keysize 2048 \
        -sigalg sha512withrsa \
```

```
        -keystore example_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

例 3: p256 ED キーを生成し、作業ディレクトリの「example_keystore.store」という名前のキーストアファイルに保存します。*<ec key pair label>* を独自のラベルに置き換えます。

```
keytool -genkeypair -alias <ec key pair label> \
        -keyalg ec -keysize 256 \
        -sigalg SHA512withECDSA \
        -keystore example_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

サポートされている署名アルゴリズムのリストは、Java ライブラリにあります。

keytool を使用して AWS CloudHSM キーを削除する

AWS CloudHSM キーストアはキーの削除をサポートしていません。キーを削除するには、 AWS CloudHSMのコマンドラインツール の deleteKey関数を使用する必要がありますKMU を使用して AWS CloudHSM キーを削除する。

keytool AWS CloudHSM を使用して CSR を生成する

AWS CloudHSM クライアント SDK 5 用の OpenSSL Dynamic Engine を使用すると、証明書署名要求 (CSR) を柔軟に生成できます。次のコマンドは、keytool を使用して、エイリアス example-key-pair を持つキーペアの CSR を生成します。

```
keytool -certreq -alias <key pair label> \
        -file example_csr.csr \
        -keystore example_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

> **Note**
>
> keytool のキーペアを使用するには、指定されたキーストアファイルにそのキーペアのエントリが必要です。keytool の外部で生成されたキーペアを使用する場合は、キーおよび証明書の

メタデータをキーストアにインポートする必要があります。キーストアデータのインポート手順については、「Keytool を使用した AWS CloudHSM キーストアへの中間証明書とルート証明書のインポート」を参照してください。

keytool を使用して中間証明書とルート証明書を AWS CloudHSM キーストアにインポートする

CA 証明書を にインポートするには AWS CloudHSM、新しくインポートされた証明書で完全な証明書チェーンの検証を有効にする必要があります。次のコマンドでは、例を示しています。

```
keytool -import -trustcacerts -alias rootCAcert \
        -file rootCAcert.cert -keystore example_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

複数のクライアントインスタンスを AWS CloudHSM クラスターに接続する場合、1 つのクライアントインスタンスのキーストアに証明書をインポートしても、他のクライアントインスタンスで証明書が自動的に使用可能になることはありません。各クライアントインスタンスで証明書をインポートする必要があります。

keytool を使用して AWS CloudHSM キーストアから証明書を削除する

次のコマンドは、Java keytool キーストアから AWS CloudHSM 証明書を削除する方法の例を示しています。

```
keytool -delete -alias mydomain -keystore \
        -keystore example_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

複数のクライアントインスタンスを AWS CloudHSM クラスターに接続しても、1 つのクライアントインスタンスのキーストアで証明書を削除しても、他のクライアントインスタンスから証明書は自動的に削除されません。各クライアントインスタンスで証明書を削除する必要があります。

keytool を使用して作業証明書を AWS CloudHSM キーストアにインポートする

証明書署名要求 (CSR) が署名されると、それを AWS CloudHSM キーストアにインポートし、適切なキーペアに関連付けることができます。次のコマンドでは、例を示しています。

```
keytool -importcert -noprompt -alias <key pair label> \
        -file example_certificate.crt \
        -keystore example_keystore.store
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

エイリアスは、キーストア内の関連付けられた証明書を持つキーペアである必要があります。キーが keytool の外部で生成される場合や、別のクライアントインスタンスで生成される場合は、まずキーおよび証明書のメタデータをキーストアにインポートする必要があります。証明書メタデータをインポートする手順については、AWS CloudHSM 「 キーストアで既存のキーを登録する」のコードサンプルを参照してください。

証明書チェーンは検証可能である必要があります。証明書を検証できない場合は、チェーンを検証できるように、署名 (証明機関) 証明書をキーストアにインポートする必要があります。

keytool AWS CloudHSM を使用して から証明書をエクスポートする

次の例では、バイナリ X.509 形式の証明書を生成します。人間が読み取り可能な証明書をエクスポートするには AWS CloudHSM、 `-exportcert` コマンド`-rfc`に を追加します。

```
keytool -exportcert -alias <key pair label> \
        -file example_exported_certificate.crt \
        -keystore example_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

クライアント SDK 3 を使用して Jarsigner で AWS CloudHSM キーストアを使用する

Jarsigner は、ハードウェア セキュリティ モジュール (HSM) に安全に保存されたキーを使用して JAR ファイルに署名するための一般的なコマンドラインユーティリティです。Jarsigner に関する完全なチュートリアルは、 AWS CloudHSM ドキュメントの対象外です。このセクションでは、 AWS CloudHSM キーストアを通じて信頼のルート AWS CloudHSM として を使用して署名および検証するために使用する Jarsigner パラメータについて説明します。

Jarsigner で AWS CloudHSM キーと証明書をセットアップする

Jarsigner で AWS CloudHSM JAR ファイルに署名する前に、次の手順を設定または完了していることを確認してください。

1. 「AWS CloudHSM キーストアの前提条件」のガイダンスに従います。
2. 現在のサーバーまたはクライアントインスタンスの AWS CloudHSM キーストアに保存する必要がある署名キー、関連する証明書、証明書チェーンを設定します。でキーを作成し AWS CloudHSM 、関連するメタデータを AWS CloudHSM キーストアにインポートします。「既存のキーを AWS CloudHSM キーストアに登録する」のコードサンプルを使用して、メタデータをキーストアにインポートします。keytool を使用してキーおよび証明書を設定する場合は、「keytool を使用して新しい AWS CloudHSM キーを作成する」を参照してください。複数のクライアントインスタンスを使用して JAR に署名する場合は、キーを作成し、証明書チェーンをインポートします。次に、結果のキーストアファイルを各クライアントインスタンスにコピーします。新しいキーを頻繁に生成する場合は、各クライアントインスタンスに証明書を個別にインポートする方が簡単です。
3. 証明書チェーン全体が検証可能である必要があります。証明書チェーンを検証できるようにするには、CA 証明書と中間証明書を AWS CloudHSM キーストアに追加する必要がある場合があります。Java コードを使用して証明書チェーンを検証する手順については、「JAR ファイルの署名」の「コードスニペット AWS CloudHSM 」と「Jarsigner」を参照してください。必要に応じて、keytool を使用して証明書をインポートできます。keytool を使用する手順については、「Keytool を使用して中間証明書とルート証明書を AWS CloudHSM Key Store にインポートする」を参照してください。

AWS CloudHSM と Jarsigner を使用して JAR ファイルに署名する

次のコマンドを使用して、 AWS CloudHSM と jarsigner を使用して JAR ファイルに署名します。

```
jarsigner -keystore example_keystore.store \
        -signedjar signthisclass_signed.jar \
        -sigalg sha512withrsa \
        -storetype CloudHSM \
        -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
        -J-Djava.library.path=/opt/cloudhsm/lib \
        signthisclass.jar <key pair label>
```

署名付き JAR を確認するには、次のコマンドを使用します。

```
jarsigner -verify \
        -keystore example_keystore.store \
        -sigalg sha512withrsa \
        -storetype CloudHSM \
        -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
        -J-Djava.library.path=/opt/cloudhsm/lib \
```

```
        signthisclass_signed.jar <key pair label>
```

クライアント SDK 3 を使用した Java Keytool と Jarsigner AWS CloudHSM の統合に関する既知の問題

次のリストは、クライアント SDK 3 を使用した AWS CloudHSM および Java Keytool および Jarsigner との統合に関する既知の問題の現在のリストを示しています。

- keytool を使用してキーを生成する場合、プロバイダー設定の最初のプロバイダを CaviumProvider にすることはできません。
- keytool を使用してキーを生成する場合、セキュリティ構成ファイル内の最初の (サポートされている) プロバイダーを使用してキーを生成します。これは通常、ソフトウェアプロバイダーです。生成されたキーにはエイリアスが割り当てられ、キーの追加プロセス中に永続的 (トークン) キーとして AWS CloudHSM HSM にインポートされます。
- keytool を AWS CloudHSM キーストアで使用するときは、コマンドラインで `-providerName`、`-providerclass`、または `-providerpath`オプションを指定しないでください。これらのオプションは、「キーストアの前提条件」の説明に従って、セキュリティプロバイダーファイルで指定します。
- keytool と Jarsigner を介して抽出不可能な EC キーを使用する場合、SunEC プロバイダーを java.security ファイル内のプロバイダーのリストから削除する、または無効にする必要があります。keytool と Jarsigner を介して抽出可能な EC キーを使用する場合、プロバイダーは AWS CloudHSM HSM からキービットをエクスポートし、キーをローカルで署名オペレーションに使用します。keytool または Jarsigner でエクスポート可能なキーを使用することはお勧めしません。

既存のキーを AWS CloudHSM キーストアに登録する

属性とラベル付けのセキュリティと柔軟性を最大限に高めるために、key_mgmt_util を使用して AWS CloudHSM 署名キーを生成することをお勧めします。Java アプリケーションを使用して、AWS CloudHSMでキーを生成することもできます。

次のセクションでは、HSM で新しいキーペアを生成し、 AWS CloudHSM キーストアにインポートされた既存のキーを使用して登録する方法を示すコードサンプルを提供します。インポートされたキーは、keytool や Jarsigner などのサードパーティー製ツールで使用できます。

既存のキーを使用するには、新しいキーを生成するのではなく、ラベルでキーを検索するようにコードサンプルを変更します。ラベルでキーを検索するためのサンプルコードは、GitHubの KeyUtilitiesRunner.java サンプルで入手できます。

> **Important**
>
> に保存されたキーを AWS CloudHSM ローカルキーストアに登録しても、キーはエクスポートされません。キーが登録されると、キーストアはキーのエイリアス (またはラベル) を登録し、 AWS CloudHSMでローカルに保存された証明書オブジェクトとのキーペアを関連付けます。キーペアがエクスポート不可として作成されている限り、キービットが HSM から離れることはありません。

```
                    //
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a copy of
this
// software and associated documentation files (the "Software"), to deal in the
Software
// without restriction, including without limitation the rights to use, copy, modify,
// merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
// permit persons to whom the Software is furnished to do so.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
// INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
// PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
// HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
// OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
// SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
//

package com.amazonaws.cloudhsm.examples;

import com.cavium.key.CaviumKey;
import com.cavium.key.parameter.CaviumAESKeyGenParameterSpec;
import com.cavium.key.parameter.CaviumRSAKeyGenParameterSpec;
import com.cavium.asn1.Encoder;
import com.cavium.cfm2.Util;

import javax.crypto.KeyGenerator;

import java.io.ByteArrayInputStream;
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;

import java.math.BigInteger;

import java.security.*;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.KeyStore.PasswordProtection;
import java.security.KeyStore.PrivateKeyEntry;
import java.security.KeyStore.Entry;

import java.util.Calendar;
import java.util.Date;
import java.util.Enumeration;

//
// KeyStoreExampleRunner demonstrates how to load a keystore, and associate a
 certificate with a
// key in that keystore.
//
// This example relies on implicit credentials, so you must setup your environment
 correctly.
//
// https://docs.aws.amazon.com/cloudhsm/latest/userguide/java-library-
install.html#java-library-credentials
//

public class KeyStoreExampleRunner {

     private static byte[] COMMON_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x03 };
     private static byte[] COUNTRY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x06 };
     private static byte[] LOCALITY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x07 };
     private static byte[] STATE_OR_PROVINCE_NAME_OID = new byte[] { (byte) 0x55,
 (byte) 0x04, (byte) 0x08 };
```

```
    private static byte[] ORGANIZATION_NAME_OID = new byte[] { (byte) 0x55, (byte)
0x04, (byte) 0x0A };
    private static byte[] ORGANIZATION_UNIT_OID = new byte[] { (byte) 0x55, (byte)
0x04, (byte) 0x0B };

    private static String helpString = "KeyStoreExampleRunner%n" +
           "This sample demonstrates how to load and store keys using a keystore.%n%n"
+
           "Options%n" +
           "\t--help\t\t\tDisplay this message.%n" +
           "\t--store <filename>\t\tPath of the keystore.%n" +
           "\t--password <password>\t\tPassword for the keystore (not your CU
password).%n" +
           "\t--label <label>\t\t\tLabel to store the key and certificate under.%n" +
           "\t--list\t\t\tList all the keys in the keystore.%n%n";

   public static void main(String[] args) throws Exception {
       Security.addProvider(new com.cavium.provider.CaviumProvider());
       KeyStore keyStore = KeyStore.getInstance("CloudHSM");

       String keystoreFile = null;
       String password = null;
       String label = null;
       boolean list = false;
       for (int i = 0; i < args.length; i++) {
           String arg = args[i];
           switch (args[i]) {
               case "--store":
                   keystoreFile = args[++i];
                   break;
               case "--password":
                   password = args[++i];
                   break;
               case "--label":
                   label = args[++i];
                   break;
               case "--list":
                   list = true;
                   break;
               case "--help":
                   help();
                   return;
           }
       }
```

```
        if (null == keystoreFile || null == password) {
            help();
            return;
        }

        if (list) {
            listKeys(keystoreFile, password);
            return;
        }

        if (null == label) {
            label = "Keystore Example Keypair";
        }

        //
        // This call to keyStore.load() will open the pkcs12 keystore with the supplied
        // password and connect to the HSM. The CU credentials must be specified using
        // standard CloudHSM login methods.
        //
        try {
            FileInputStream instream = new FileInputStream(keystoreFile);
            keyStore.load(instream, password.toCharArray());
        } catch (FileNotFoundException ex) {
            System.err.println("Keystore not found, loading an empty store");
            keyStore.load(null, null);
        }

        PasswordProtection passwd = new PasswordProtection(password.toCharArray());
        System.out.println("Searching for example key and certificate...");

        PrivateKeyEntry keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
        if (null == keyEntry) {
            //
            // No entry was found, so we need to create a key pair and associate a
certificate.
            // The private key will get the label passed on the command line. The
keystore alias
            // needs to be the same as the private key label. The public key will have
":public"
            // appended to it. The alias used in the keystore will We associate the
certificate
            // with the private key.
            //
```

```
            System.out.println("No entry found, creating...");
            KeyPair kp = generateRSAKeyPair(2048, label + ":public", label);
            System.out.printf("Created a key pair with the handles %d/%d%n",
((CaviumKey) kp.getPrivate()).getHandle(), ((CaviumKey) kp.getPublic()).getHandle());

            //
            // Generate a certificate and associate the chain with the private key.
            //
            Certificate self_signed_cert = generateCert(kp);
            Certificate[] chain = new Certificate[1];
            chain[0] = self_signed_cert;
            PrivateKeyEntry entry = new PrivateKeyEntry(kp.getPrivate(), chain);

            //
            // Set the entry using the label as the alias and save the store.
            // The alias must match the private key label.
            //
            keyStore.setEntry(label, entry, passwd);

            FileOutputStream outstream = new FileOutputStream(keystoreFile);
            keyStore.store(outstream, password.toCharArray());
            outstream.close();

            keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
        }

        long handle = ((CaviumKey) keyEntry.getPrivateKey()).getHandle();
        String name = keyEntry.getCertificate().toString();
        System.out.printf("Found private key %d with certificate %s%n", handle, name);
    }

    private static void help() {
        System.out.println(helpString);
    }

    //
    // Generate a non-extractable / non-persistent RSA keypair.
    // This method allows us to specify the public and private labels, which
    // will make KeyStore aliases easier to understand.
    //
    public static KeyPair generateRSAKeyPair(int keySizeInBits, String publicLabel,
String privateLabel)
            throws InvalidAlgorithmParameterException, NoSuchAlgorithmException,
NoSuchProviderException {
```

```
        boolean isExtractable = false;
        boolean isPersistent = false;
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
        CaviumRSAKeyGenParameterSpec spec = new
CaviumRSAKeyGenParameterSpec(keySizeInBits, new BigInteger("65537"), publicLabel,
privateLabel, isExtractable, isPersistent);

        keyPairGen.initialize(spec);

        return keyPairGen.generateKeyPair();
    }

    //
    // Generate a certificate signed by a given keypair.
    //
    private static Certificate generateCert(KeyPair kp) throws CertificateException {
        CertificateFactory cf = CertificateFactory.getInstance("X509");
        PublicKey publicKey = kp.getPublic();
        PrivateKey privateKey = kp.getPrivate();
        byte[] version = Encoder.encodeConstructed((byte) 0,
Encoder.encodePositiveBigInteger(new BigInteger("2"))); // version 1
        byte[] serialNo = Encoder.encodePositiveBigInteger(new BigInteger(1,
Util.computeKCV(publicKey.getEncoded())));

        // Use the SHA512 OID and algorithm.
        byte[] signatureOid = new byte[] {
            (byte) 0x2A, (byte) 0x86, (byte) 0x48, (byte) 0x86, (byte) 0xF7, (byte)
0x0D, (byte) 0x01, (byte) 0x01, (byte) 0x0D };
        String sigAlgoName = "SHA512WithRSA";

         byte[] signatureId = Encoder.encodeSequence(
                                        Encoder.encodeOid(signatureOid),
                                        Encoder.encodeNull());

         byte[] issuer = Encoder.encodeSequence(
                                    encodeName(COUNTRY_NAME_OID, "<Country>"),
                                    encodeName(STATE_OR_PROVINCE_NAME_OID, "<State>"),
                                    encodeName(LOCALITY_NAME_OID, "<City>"),
                                    encodeName(ORGANIZATION_NAME_OID,
"<Organization>"),
                                    encodeName(ORGANIZATION_UNIT_OID, "<Unit>"),
                                    encodeName(COMMON_NAME_OID, "<CN>")
                                );
```

```
        Calendar c = Calendar.getInstance();
        c.add(Calendar.DAY_OF_YEAR, -1);
        Date notBefore = c.getTime();
        c.add(Calendar.YEAR, 1);
        Date notAfter = c.getTime();
        byte[] validity = Encoder.encodeSequence(
                                    Encoder.encodeUTCTime(notBefore),
                                    Encoder.encodeUTCTime(notAfter)
                                );
        byte[] key = publicKey.getEncoded();

        byte[] certificate = Encoder.encodeSequence(
                                    version,
                                    serialNo,
                                    signatureId,
                                    issuer,
                                    validity,
                                    issuer,
                                    key);
        Signature sig;
        byte[] signature = null;
        try {
            sig = Signature.getInstance(sigAlgoName, "Cavium");
            sig.initSign(privateKey);
            sig.update(certificate);
            signature = Encoder.encodeBitstring(sig.sign());

        } catch (Exception e) {
            System.err.println(e.getMessage());
            return null;
        }

        byte [] x509 = Encoder.encodeSequence(
                        certificate,
                        signatureId,
                        signature
                        );
        return cf.generateCertificate(new ByteArrayInputStream(x509));
    }

     //
     // Simple OID encoder.
     // Encode a value with OID in ASN.1 format
```

```
     //
     private static byte[] encodeName(byte[] nameOid, String value) {
         byte[] name = null;
         name = Encoder.encodeSet(
                     Encoder.encodeSequence(
                             Encoder.encodeOid(nameOid),
                             Encoder.encodePrintableString(value)
                     )
                 );
         return name;
     }

    //
    // List all the keys in the keystore.
    //
    private static void listKeys(String keystoreFile, String password) throws Exception
 {
        KeyStore keyStore = KeyStore.getInstance("CloudHSM");

        try {
            FileInputStream instream = new FileInputStream(keystoreFile);
            keyStore.load(instream, password.toCharArray());
        } catch (FileNotFoundException ex) {
            System.err.println("Keystore not found, loading an empty store");
            keyStore.load(null, null);
        }

        for(Enumeration<String> entry = keyStore.aliases(); entry.hasMoreElements();) {
            System.out.println(entry.nextElement());
        }
    }

}
```

で Microsoft Manifest Generation and Editing Tool (Mage.exe) AWS CloudHSM を使用してファイルに署名する

> **Note**
>
> AWS CloudHSM は、Windows SDK for .NET Framework 4.8.1 以降に含まれている 64 ビット Mage ツールのみをサポートします。

以下のトピックでは、 で Mage.exe を使用する方法の概要を説明します AWS CloudHSM。

トピック

- ステップ 1: 前提条件の設定
- ステップ 2: 署名用証明書を作成する
- ステップ 3: ファイルに署名する

ステップ 1: 前提条件の設定

で Microsoft Mage.exe を使用するには AWS CloudHSM、以下が必要です。

- Windows オペレーティングシステムが実行されている Amazon EC2 インスタンス。
- 自前で管理する、またはサードパーティプロバイダーによる認証局 (CA)
- EC2 インスタンスと同じ Virtual Private Cloud (VPC) に少なくとも 1 つの HSM があるアクティブな AWS CloudHSM クラスター
- AWS CloudHSM クラスター内のキーを所有および管理するための Crypto User (CU)
- 未署名のファイルまたは実行可能ファイル
- Microsoft Windows Software Development Kit (SDK)

Mage.exe AWS CloudHSM で を使用するための前提条件を設定するには

1. このガイドの「開始方法???」セクションの手順に従って、Windows EC2 インスタンスと AWS CloudHSM クラスターを起動します。
2. 独自の Windows Server CA をホストする場合は、「 で Windows Server を認証機関として設定 AWS CloudHSMする」のステップ 1 と 2 を完了します。それ以外の場合は、公的に信頼されたサードパーティーの CA を使用します。

3. .NET Framework 4.8.1 以降向けの Microsoft Windows SDK を Windows EC2 インスタンスにダウンロードしてインストールします。

   - Microsoft Windows SDK 10

   `mage.exe` 実行可能ファイルは Windows SDK ツールの一部です。デフォルトのインストール場所は次のとおりです。

   ```
   C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\x64\Mage.exe
   ```

これらのステップを完了したら、Microsoft Windows SDK、 AWS CloudHSM クラスター、および CA を使用して署名証明書を作成できます。

ステップ 2: 署名用証明書を作成する

EC2 インスタンスに Windows SDK をインストールしたら、これを使用して証明書署名リクエスト (CSR) を生成することができます。CSR は、未署名の証明書であり、署名用に最終的に CA に提出します。この例では、Windows SDK に含まれる `certreq` 実行可能ファイルを使用して、CSR を生成します。

certreq 実行可能ファイルを使用して CSR を生成するには

1. Windows EC2 インスタンスに接続します。詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスに接続する」を参照してください。
2. `request.inf` という名前のファイルを作成し、次の内容を記述します。`Subject` 情報をお客様の組織の情報に置き換えます。

   ```
   [Version]
   Signature= $Windows NT$
   [NewRequest]
   Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-
   Unit>,L=<City>,S=<State>"
   RequestType=PKCS10
   HashAlgorithm = SHA256
   KeyAlgorithm = RSA
   KeyLength = 2048
   ProviderName = "CloudHSM Key Storage Provider"
   KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
   MachineKeySet = True
   ```

```
Exportable = False
```

各パラメータの説明については、「Microsoft のドキュメント」を参照してください。

3. certreq.exe を実行して CSR を生成します。

```
certreq.exe -new request.inf request.csr
```

このコマンドは、 AWS CloudHSM クラスターに新しいキーペアを生成し、プライベートキーを使用して CSR を作成します。

4. CA に CSR を送ります。Windows Server CA を使用している場合は、次のステップを行います。

   a. CA ツールを開きます。

   ```
   certsrv.msc
   ```

   b. 新しいウィンドウで、CA サーバーの名前を右クリックします。[すべてのタスク]、[Submit new request (新しいリクエストの送信)] の順に選択します。

   c. request.csr の場所に移動し、[開く] を選択します。

   d. [サーバー CA] メニューから、[保留中のリクエスト] フォルダを表示します。先ほど作成したリクエストを右クリックし、[すべてのタスク] を選択し、 [問題] を選択します。

   e. [発行された証明書] フォルダに移動します。

   f. [開く] を選択して証明書を表示し、[詳細] タブを選択します。

   g. [Copy to File (ファイルにコピー)] を選択して、証明書のエクスポートウィザードを起動します。DER でエンコードされた X.509 ファイルを signedCertificate.cer として安全な場所に保存します。

   h. CA ツールを終了し、次のコマンドを実行して、証明書ファイルを Windows の Personal Certificate Store に移動します。

   ```
   certreq.exe -accept signedCertificate.cer
   ```

これで、インポートしたファイルを使用して、ファイルに署名することができます。

ステップ 3: ファイルに署名する

Mage.exe とインポートした証明書を取得したため、ファイルに署名できます。証明書の SHA-1 ハッシュ (サムプリント) を把握しておく必要があります。サムプリントにより、Mage.exe は AWS CloudHSMによって検証された証明書のみを使用します。この例では、PowerShell を使用して証明書のハッシュを取得します。

証明書のサムプリントを取得し、それを使用してファイルに署名するには

1. mage.exe が含まれているディレクトリに移動します。デフォルトの場所:

```
C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8.1 Tools\x64
```

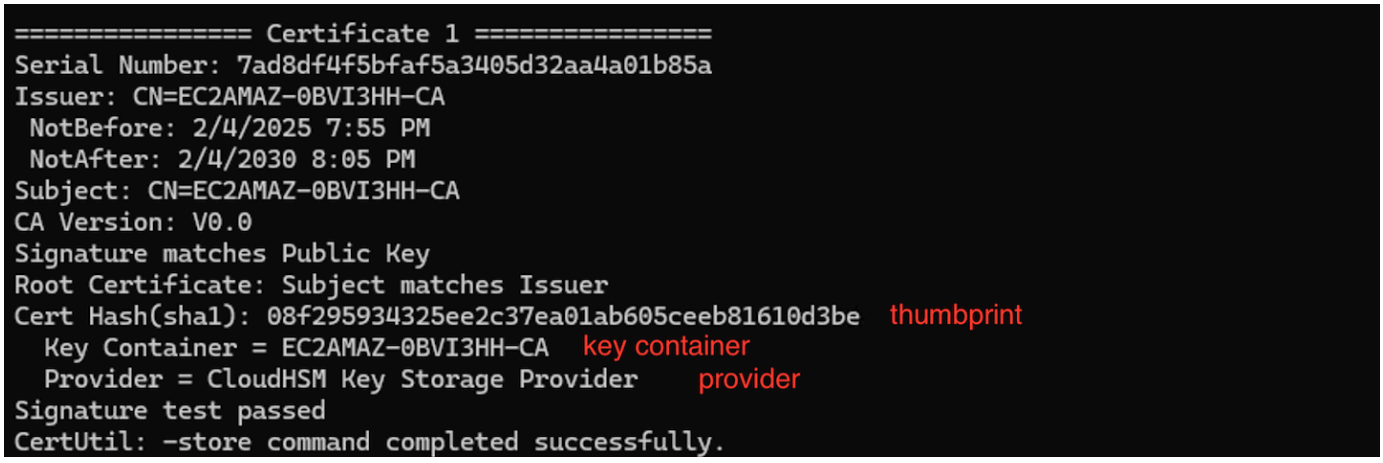
2. Mage.exe を使用してサンプルアプリケーションファイルを作成するには、次のコマンドを実行します。

```
mage.exe -New Application -ToFile C:\Users\Administrator\Desktop\sample.application
```

3. 管理者として PowerShell を開き、次のコマンドを実行します。

```
Get-ChildItem -path cert:\LocalMachine\My
```

出力から Thumbprint、Key Container、Provider の値をコピーします。



4. 次のコマンドを実行してファイルに署名します。

```
mage.exe -Sign -CertHash <thumbprint> -KeyContainer <keycontainer> -
CryptoProvider <CloudHSM Key Storage Provider/Cavium Key Storage Provider> C:\Users
\Administrator\Desktop\<sample.application>
```

コマンドが正常に実行されると、PowerShell より成功のメッセージが返ります。

5. ファイルの署名を検証するには、次のコマンドを使用します。

```
mage.exe -Verify -CryptoProvider <CloudHSM Key Storage Provider/Cavium Key Storage
 Provider> C:\Users\Administrator\Desktop\<sample.application>
```

とのその他のサードパーティーベンダー統合 AWS CloudHSM

一部のサードパーティーベンダーは、信頼のルート AWS CloudHSM として をサポートしています。これはつまり、CloudHSM クラスターで基になるキーを作成、保存しながら、選択したソフトウェアソリューションを利用できるということです。その結果、 のワークロードはCloudHSM のレイテンシー、可用性、信頼性、伸縮性の利点に依存する AWS 可能性があります。次の表は、CloudHSM をサポートするサードパーティベンダーのリストです。

> **Note**
>
> AWS は、サードパーティーベンダーを支持または保証しません。

- Hashicorp Vault は、組織間のコラボレーションおよびガバナンスを可能にするために設計されたシークレット管理ツールです。保護を強化するために、信頼のルート AWS CloudHSM として AWS Key Management Service と をサポートしています。
- Thycotic Secrets Server を使用すると、機密性の高い認証情報を特権アカウント間で管理できます。信頼のルート AWS CloudHSM として をサポートします。
- P6R の KMIP アダプターを使用すると、標準の KMIP インターフェイスを介して AWS CloudHSM インスタンスを利用できます。
- PrimeKey EJBCA は、PKI 用の一般的なオープンソースソリューションです。これにより、 でキーペアを安全に作成して保存できます AWS CloudHSM。
- Box KeySafe は、厳格なセキュリティやプライバシー、法令遵守の要件を持つ多くの組織にクラウドコンテンツの暗号化キー管理を提供します。お客様は、カスタムキーストアを介して で直接、または で間接的に KeySafe AWS KMS キーをさらに保護できます。 AWS Key Management Service AWS CloudHSM
- Insyde Software は、ファームウェア署名の信頼のルート AWS CloudHSM として をサポートします。

- F5 BIG-IP LTM は、信頼のルート AWS CloudHSM として をサポートします。
- Cloudera Navigator Key HSM を使用すると、CloudHSM クラスターを使用してCloudera Navigator Key Trustee Server のキーを作成および保存できます。
- Venafi 信頼保護プラットフォーム は、AWS CloudHSM キー生成と保護により、TLS、SSH、およびコードサイニングのための包括的なマシン ID 管理を提供します。

モニタリング AWS CloudHSM

Client SDK に組み込まれているログ記録機能に加えて、 AWS CloudTrail、Amazon CloudWatch Logs、Amazon CloudWatch を使用してモニタリングすることもできます AWS CloudHSM。

クライアント SDK ログ

: クライアント SDK ロギングを使用して、作成したアプリケーションからの診断およびトラブルシューティング情報をモニタリングします。

CloudTrail

: CloudTrail を使用して、クラスターの作成と削除のための呼び出し、ハードウェアセキュリティモジュール (HSM)、リソースタグなど、 AWS アカウント内のすべての API コールをモニタリングします。

CloudWatch ログ

: CloudWatch Logs を使用して、HSM インスタンスからのログをモニタリングします。これには、HSM ユーザーを作成及び削除、ユーザーパスワードの変更、キーの作成および削除などのイベントが含まれます。

CloudWatch

: CloudWatch を使用して、リアルタイムでクラスターのヘルスのモニタリングを行います。

トピック

- AWS CloudHSM クライアント SDK ログの使用
- AWS CloudTrailおよび の使用AWS CloudHSM
- Amazon CloudWatch Logs と AWS CloudHSM 監査ログの使用
- の CloudWatch メトリクスの取得 AWS CloudHSM

AWS CloudHSM クライアント SDK ログの使用

クライアント SDK によって生成されたログを取得できます。 は、クライアント SDK 3 およびクライアント SDK 5 を使用したログ記録の実装 AWS CloudHSM を提供します。

トピック

- クライアント SDK 5 ログ記録
- クライアント SDK 3 ログ記録

クライアント SDK 5 ログ記録

クライアント SDK 5 ログには、コンポーネントのために名付けられたファイル中の各コンポーネントのための情報が含まれています。クライアント SDK 5 の設定ツールを使用して、各コンポーネントのログを構成できます。

ファイルの場所を指定しない場合、システムはログをデフォルトの場所に書き込みます。

AWS CloudHSM CLI

- Linux

  ```
  /opt/cloudhsm/run/cloudhsm-cli.log
  ```

  Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
  ```

PKCS #11 library

- Linux

  ```
  /opt/cloudhsm/run/cloudhsm-pkcs11.log
  ```

  Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
  ```

OpenSSL Dynamic Engine

- Linux

  ```
  stderr
  ```

JCE provider

- Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```

Key Storage Provider

- Server

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-ksp.log
  ```

OpenSSL Dynamic Engine Provider

- Linux

  ```
  stderr
  ```

クライアント SDK 5 のログ記録を構成する方法については、「クライアント SDK 5 Configure tool」を参照してください

クライアント SDK 3 ログ記録

クライアント SDK 3 ログには、 AWS CloudHSM クライアントデーモンの詳細情報が含まれています。ログの場所は、クライアントデーモンを実行するクライアントを実行する Amazon EC2 クライアントインスタンスのオペレーティングシステムによって異なります。

Amazon Linux

Amazon Linux では、 AWS CloudHSM クライアントログは という名前のファイルに書き込まれます`/opt/cloudhsm/run/cloudhsm_client.log`。logrotate などを使用して、これらのログをローテーションして管理します。

Amazon Linux 2

Amazon Linux 2 では、 AWS CloudHSM クライアントログが収集され、ジャーナルに保存されます。journalctl を使用して、これらのログを表示および管理できます。たとえば、次のコマンドを使用して AWS CloudHSM クライアントログを表示します。

```
journalctl -f -u cloudhsm-client
```

CentOS 7

CentOS 7 では、 AWS CloudHSM クライアントログが収集され、ジャーナルに保存されます。journalctl を使用して、これらのログを表示および管理できます。たとえば、次のコマンドを使用して AWS CloudHSM クライアントログを表示します。

```
journalctl -f -u cloudhsm-client
```

CentOS 8

CentOS 8 では、 AWS CloudHSM クライアントログが収集され、ジャーナルに保存されます。journalctl を使用して、これらのログを表示および管理できます。たとえば、次のコマンドを使用して AWS CloudHSM クライアントログを表示します。

```
journalctl -f -u cloudhsm-client
```

RHEL 7

Red Hat Enterprise Linux 7 では、 AWS CloudHSM クライアントログが収集され、ジャーナルに保存されます。journalctl を使用して、これらのログを表示および管理できます。たとえば、次のコマンドを使用して AWS CloudHSM クライアントログを表示します。

```
journalctl -f -u cloudhsm-client
```

RHEL 8

Red Hat Enterprise Linux 8 では、 AWS CloudHSM クライアントログが収集され、ジャーナルに保存されます。journalctl を使用して、これらのログを表示および管理できます。たとえば、次のコマンドを使用して AWS CloudHSM クライアントログを表示します。

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

Ubuntu 16.04 では、 AWS CloudHSM クライアントログが収集され、ジャーナルに保存されます。journalctl を使用して、これらのログを表示および管理できます。たとえば、次のコマンドを使用して AWS CloudHSM クライアントログを表示します。

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

Ubuntu 18.04 では、 AWS CloudHSM クライアントログが収集され、ジャーナルに保存されます。journalctl を使用して、これらのログを表示および管理できます。たとえば、次のコマンドを使用して AWS CloudHSM クライアントログを表示します。

```
journalctl -f -u cloudhsm-client
```

Windows

- Windows クライアント 1.1.2+ の場合:

  AWS CloudHSM クライアントログは、プログラム`cloudhsm-kps.log`ファイルフォルダ () のファイルに書き込まれます AWS CloudHSM `C:\Program Files\Amazon\CloudHSM\`。各ログファイル名には、 AWS CloudHSM クライアントがいつ起動されたかを示すタイムスタンプが付きます。
- Windows クライアント 1.1.1 以前の場合:

  クライアントのログはファイルに書き込まれません。ログは、コマンドプロンプトまたは AWS CloudHSM クライアントを開始した PowerShell ウィンドウに表示されます。

AWS CloudTrailおよび の使用AWS CloudHSM

AWS CloudHSMは、ユーザーAWS CloudTrail、ロール、または のサービスによって実行されたアクションを記録するAWSサービスである と統合されていますAWS CloudHSM。CloudTrail は、 のすべての API コールをイベントAWS CloudHSMとしてキャプチャします。キャプチャされた呼び出しには、AWS CloudHSMコンソールからの呼び出しと AWS CloudHSMAPI オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、イベントを含む Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができますAWS CloudHSM。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail

で収集された情報を使用して、リクエストの実行元の IP アドレスAWS CloudHSM、リクエストの実行者、リクエストの実行日時などの詳細を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。AWS CloudHSMAPI オペレーションの完全なリストについては、AWS CloudHSM「 API リファレンス」の「アクション」を参照してください。

AWS CloudHSMCloudTrail の情報

CloudTrail は、AWSアカウントの作成時にアカウントで有効になります。でアクティビティが発生するとAWS CloudHSM、そのアクティビティはイベント履歴の他のAWSサービスイベントとともに CloudTrail イベントに記録されます。最近のイベントは、AWS アカウントで表示、検索、ダウンロードできます。詳細については、CloudTrail イベント履歴でのイベントの表示を参照してください。

のイベントなど、 AWSアカウントのイベントの継続的な記録についてはAWS CloudHSM、証跡を作成します。追跡により、CloudTrail はログファイルを Simple Storage Service (Amazon S3) バケットに配信できます。デフォルトでは、コンソールで作成した証跡がすべての AWS リージョンに適用されます。証跡は、AWSパーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWSサービスを設定できます。詳細については、次を参照してください:

- 証跡の作成のための概要
- CloudTrail がサポートするサービスと統合
- CloudTrail 用 Amazon SNS 通知の構成
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

CloudTrail は、 `DescribeClusters`や などの読み取り専用AWS CloudHSMオペレーション、、 `ListTags`などの管理オペレーションを含むすべてのオペレーションInitializeClusterCreatHsmを`DeleteBackup`ログに記録します。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management(IAM) ユーザー認証情報を使用して行われたかどうか。

- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別のAWSサービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

AWS CloudHSMログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは、任意の出典からの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどに関する情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、 `CreateHsm`アクションを示す CloudTrail AWS CloudHSMログエントリを示しています。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAJZVM5NEGZSTCITAMM:ExampleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/AdminRole/ExampleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIY22AX6VRYNBGJSA",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2017-07-11T03:48:44Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAJZVM5NEGZSTCITAMM",
                "arn": "arn:aws:iam::111122223333:role/AdminRole",
                "accountId": "111122223333",
                "userName": "AdminRole"
            }
        }
    },
    "eventTime": "2017-07-11T03:50:45Z",
    "eventSource": "cloudhsm.amazonaws.com",
```

```
    "eventName": "CreateHsm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.179",
    "userAgent": "aws-internal/3",
    "requestParameters": {
        "availabilityZone": "us-west-2b",
        "clusterId": "cluster-fw7mh6mayb5"
    },
    "responseElements": {
        "hsm": {
            "eniId": "eni-65338b5a",
            "clusterId": "cluster-fw7mh6mayb5",
            "state": "CREATE_IN_PROGRESS",
            "eniIp": "10.0.2.7",
            "hsmId": "hsm-6lz2hfmnzbx",
            "subnetId": "subnet-02c28c4b",
            "availabilityZone": "us-west-2b"
        }
    },
    "requestID": "1dae0370-65ec-11e7-a770-6578d63de907",
    "eventID": "b73a5617-8508-4c3d-900d-aa8ac9b31d08",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


AWS CloudHSMCloudTrail の管理イベント

管理イベントは、 のリソースで実行される管理オペレーションに関する情報を提供しますAWS アカウント。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。CloudTrail は、デフォルトで管理イベントをログ記録します。

AWS CloudHSMは、すべてのコントロールプレーンオペレーションを管理イベントとしてログに記録します。CloudTrail にAWS CloudHSMログ記録するコントロールプレーンオペレーションのリストについては、 AWS CloudHSMAPI リファレンスのAWS CloudHSM「アクション」を参照してください。

AWS CloudHSMCloudTrail のサービスイベント

AWS CloudHSMは、サービスがリソースに対して自動的にアクションを実行するときに、サービスイベントを CloudTrail に記録します。これらのイベントは、パブリック AWS CloudHSMAPI への呼び出しによって直接トリガーされない場合があります。これらのイベントの場合、`eventType` フィールドは `AwsServiceEvent` です。

以下は、CloudTrail にAWS CloudHSMログ記録されるサービスイベントです。

- `DeleteBackup` – がバックアップ保持ポリシーに基づいてバックアップAWS CloudHSMを自動的に削除するとログに記録されます。
- `BackupDeletedPermanently` — PENDING_DELETION バックアップが完全に消去され、復元できなくなったときにログに記録されます。
- `MigrationStart` – クラスターの HSM タイプの移行が開始されるとログに記録されます。
- `MigrationEnd` – クラスターの HSM タイプの移行が完了するとログに記録されます。
- `RollbackStart` – クラスターの HSM タイプの移行ロールバックが開始されるとログに記録されます。
- `RollbackEnd` – クラスターの HSM タイプの移行ロールバックが完了するとログに記録されます。

CloudTrail ログのサービスAWS CloudHSMイベントは、次の属性値で識別できます。

- `eventSource`: `cloudhsm.amazonaws.com`
- `eventType`: `AwsServiceEvent`

Amazon CloudWatch Logs と AWS CloudHSM 監査ログの使用

アカウントの HSM は、コマンド AWS CloudHSM ラインツールまたはソフトウェアライブラリからコマンドを受信すると、コマンドの実行を監査ログフォームに記録します。HSM の監査ログには、HSM の作成/削除、HSM へのログイン/ログアウト、ユーザーおよびキーの管理を含む、クライアント初期化のすべての管理コマンドが含まれます。このログは、HSM の状態を変更したアクションの信頼性のある記録を提供します。

AWS CloudHSM は HSM 監査ログを収集し、ユーザーに代わって Amazon CloudWatch Logs に送信します。CloudWatch Logs の機能を使用して、ログの検索とフィルタリング、Amazon S3 へのログデータのエクスポートなど、 AWS CloudHSM 監査ログを管理できます。[Amazon CloudWatch コンソール] のHSM 監査ログまたは、[AWS CLI] や [CloudWatch Logs SDK] のCloudWatch Logs コマンドを使用して操作します。

トピック

- HSM 監査ログの記録の仕組み
- CloudWatch Logs での AWS CloudHSM 監査ログの表示

- AWS CloudHSM 監査ログの解釈
- AWS CloudHSM 監査ログリファレンス

HSM 監査ログの記録の仕組み

監査ログ記録は、すべての AWS CloudHSM クラスターで自動的に有効になります。無効にしたり無効にしたりすることはできません。また、 がログを CloudWatch Logs にエクスポート AWS CloudHSM できない設定はありません。各ログイベントには、タイムスタンプとイベントの順序を示すシーケンス番号があり、ログの改ざんを検出するために役立ちます。

HSM インスタンスごとに独自のログが生成されます。別々の HSM イベントの監査ログは、同じクラスターにある場合でも異なることがあります。たとえば、各クラスターの最初の HSM のみが HSM の初期化を記録します。初期化イベントは、バックアップからクローンされた HSM のログ記録には表示されません。同様に、キーを作成する場合、キーを生成する HSM がキーの生成イベントを記録します。クラスターの別の HSM は、同期を介してキーを受け取るときに、イベントを記録します。

AWS CloudHSM はログを収集し、アカウントの CloudWatch Logs に投稿します。ユーザーに代わって CloudWatch Logs サービスと通信するために、 はサービスにリンクされたロール AWS CloudHSM を使用します。ロールに関連付けられている IAM ポリシーにより AWS CloudHSM 、 は監査ログを CloudWatch Logs に送信するために必要なタスクのみを実行できます。

> **Important**
>
> 2018 年 1 月 20 日以前にクラスターを作成し、サービスにリンクされたロールをまだ作成していない場合には、手動でこのロールを 1 つ作成する必要があります。これは、CloudWatch が AWS CloudHSM クラスターから監査ログを受信するために必要です。サービスにリンクされたロールの作成の詳細については、[サービスにリンクされたロールを理解する] と、[IAM ユーザーガイド] の [サービスにリンクされたロールを作成する] を参照してください。

CloudWatch Logs での AWS CloudHSM 監査ログの表示

Amazon CloudWatch Logs は、監査ログを [ロググループ] と、ロググループ内で [ログストリーム] に作成します。各ログエントリは event. AWS CloudHSM creates one log group for each cluster and one log stream for each HSM in the cluster。CloudWatch Logs コンポーネントを作成する必要も、設定を変更する必要もありません。

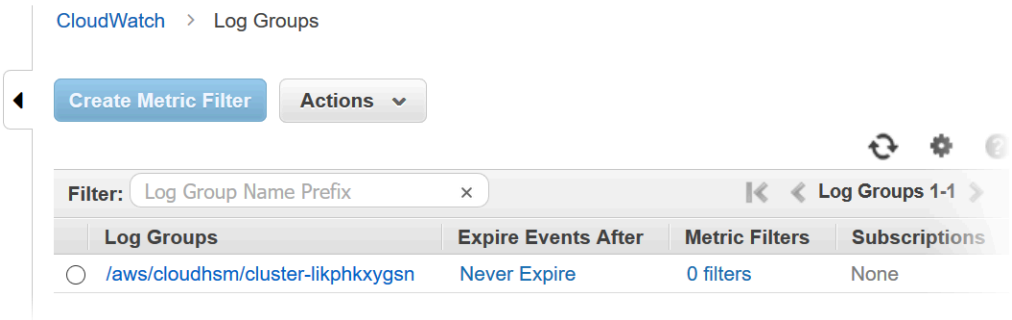
- ロググループの名前は /aws/cloudhsm/*<cluster ID>* です (たとえば、/aws/cloudhsm/cluster-likphkxygsn)。 AWS CLI あるいは PowerShell コマンドでロググループを使用する場合、必ずこれを二重引用符で囲んでください。
- ログストリーム名は HSM ID です (たとえば、hsm-nwbbiqbj4jk)。

  一般的には、各 HSM に 1 つのログストリームがあります。ただし、HSM ID を変更するすべてのアクション (HSM が失敗して置き換えられた場合など) は新しいログストリームを作成します。

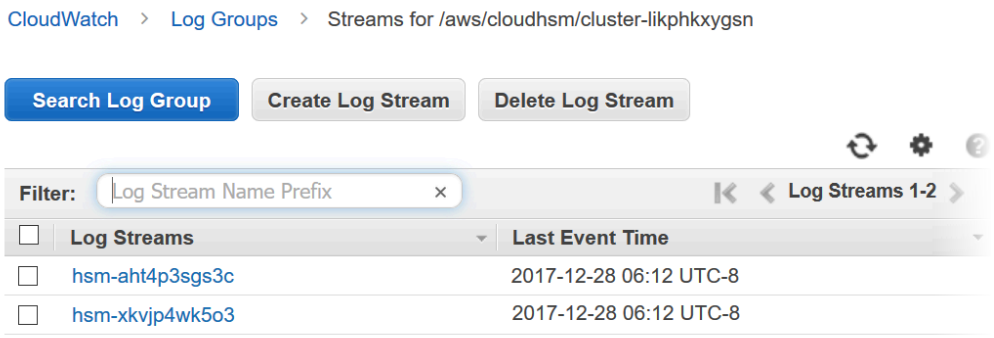
CloudWatch Logs コンセプトの詳細については、[Amazon CloudWatch Logs ユーザーガイド] の[概念] を参照してください。

HSM の監査ログは、 の CloudWatch Logs ページ AWS マネジメントコンソール、 の CloudWatch Logs コマンド AWS CLI、CloudWatch Logs PowerShell コマンドレット、または CloudWatch Logs SDKs から表示できます。手順については、[Amazon CloudWatch Logs ユーザーガイド] の [ログデータの表示] を参照してください。

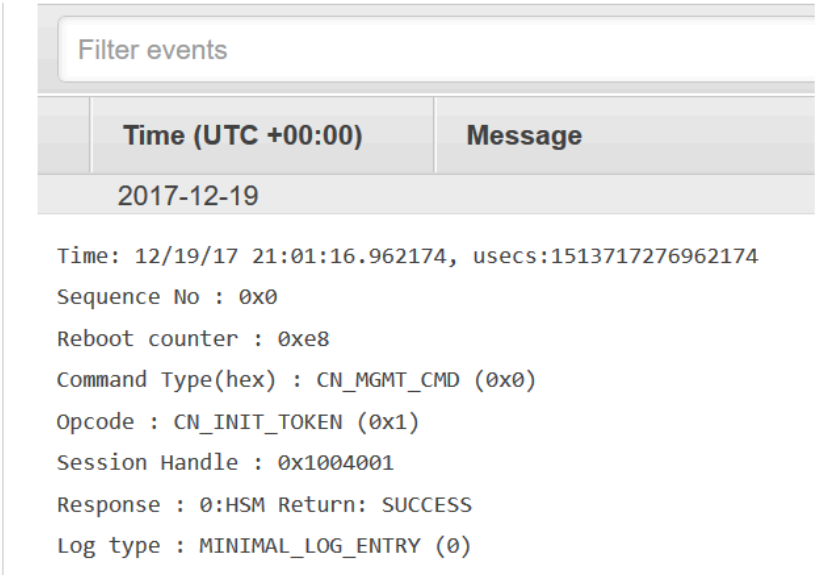
たとえば、次の図は AWS マネジメントコンソールの cluster-likphkxygsn クラスターのロググループを示しています。



クラスターのロググループ名を選択すると、このクラスターの各 HSM のログストリームを表示することができます。つぎの図は、cluster-likphkxygsn クラスターの HSM のログストリームを示しています。

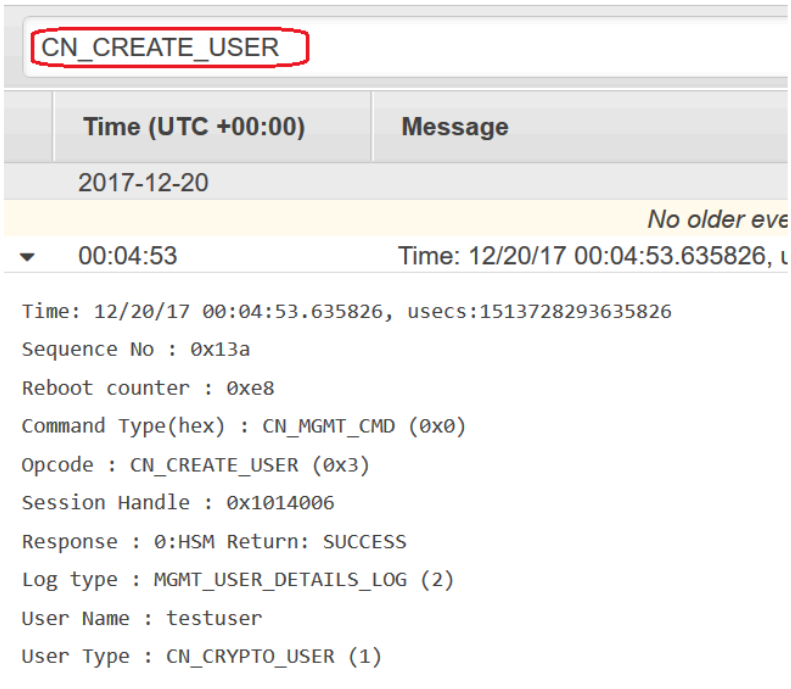


HSM のログストリーム名を選択すると、監査ログのイベントを表示することができます。たとえば、0x0 のシーケンス番号と CN_INIT_TOKEN の Opcode があるこのイベントは、通常の場合、各クラスターの最初の HSM の最初のイベントです。これには、このクラスターで HSM が初期化されたことが記録されています。



CloudWatch Logs にある多くの機能を使用して、監査ログを管理できます。たとえば、イベントのフィルター機能を使用すると、イベント内で特定のテキスト (CN_CREATE_USER Opcode など) を検索できます。

特定のテキストを含まないすべてのイベントを検索するには、テキストの前に負符号 (-) を追加します。たとえば、CN_CREATE_USER を含まないイベントを見つけるには、-CN_CREATE_USER を入力します。



AWS CloudHSM 監査ログの解釈

HSM 監査ログ内のイベントには、標準フィールドがあります。一部のイベントタイプには、イベントに関する有益な情報をキャプチャする追加のフィールドがあります。たとえば、ユーザーログインおよびユーザー管理イベントには、ユーザーのユーザー名とユーザータイプが含まれています。キー管理コマンドには、キーハンドルが含まれます。

いくつかのフィールドは、特に重要な情報を提供しています。Opcode は、記録している管理コマンドを識別します。Sequence No は、イベントをログストリームで識別して、記録された順序を示します。

たとえば、次のログイベント例は、HSM のログストリームで 2 番目のイベント (Sequence No: 0x1) です。これには、HSM が生成するパスワード暗号化キーが示され、これは起動ルーチンの一部です。

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
```

```
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

次のフィールドは、監査ログのすべての AWS CloudHSM イベントに共通です。

Time

イベントが発生した時間 (UTC タイムゾーン)。この時間は、人間が読み取れる時間と Unix 時間 (マイクロ秒) で表示されます。

再起動カウンタ

HSM ハードウェアが再起動されたときに増加する、32 ビットの永続的な序数カウンタ。

ログストリームのすべてのイベントには、同じ再起動カウンタ値があります。ただし、再起動カウンタは同じクラスターの別々の HSM インスタンスでは異なることがあるため、このカウンタはログストリームに固有ではないことがあります。

シーケンスなし

ログイベントごとに増加する 64 ビット序数カウンタ。各ログストリームの最初のイベントのシーケンス番号は 0 x 0 です。Sequence No 値でギャップがないようにする必要があります。シーケンス番号は、ログストリーム内でのみ一意です。

コマンドタイプ

コマンドのカテゴリを示す 16 進値です。 AWS CloudHSM ログストリームのコマンドには、CN_MGMT_CMD (0x0) あるいは CN_CERT_AUTH_CMD (0x9) のコマンドタイプがあります。

Opcode

実行された管理コマンドを識別します。 AWS CloudHSM 監査ログ0pcodeの値のリストについては、「」を参照してくださいAWS CloudHSM 監査ログリファレンス。

セッションハンドル

コマンドが実行され、イベントがログされたセッションを識別します。

応答

レスポンスを管理コマンドに記録します。Response フィールドで SUCCESS および ERROR 値を検索できます。

ログタイプ

コマンドを記録したログの AWS CloudHSM ログタイプを示します。

- MINIMAL_LOG_ENTRY (0)
- MGMT_KEY_DETAILS_LOG (1)
- MGMT_USER_DETAILS_LOG (2)
- GENERIC_LOG

監査ログイベントの例

ログストリーム内のイベントには、作成から削除までの HSM の履歴が記録されます。ログを使用して HSM のライフサイクルを確認し、オペレーションを把握することができます。イベントを解釈するときに、管理コマンドあるいはアクションを示す Opcode とイベントの順序を示す Sequence No に注目します。

トピック

- 例: クラスターの最初の HSM を初期化する
- ログインとログアウトイベント
- 例: ユーザーの作成と削除
- 例: キーペアの作成と削除
- 例: キーの生成と同期
- 例: キーのエクスポート
- 例: キーのインポート
- 例: キーの共有と共有解除

例: クラスターの最初の HSM を初期化する

各クラスターの最初の HSM の監査ログストリームは、クラスターの他の HSM のログストリームとは大幅に異なります。各クラスターの最初の HSM の監査ログは、この HSM の作成と初期化を記録します。クラスターに追加する HSM のログはバックアップから生成され、ログインイベントで始まります。

> **Important**
>
> 以下の初期化エントリは、CloudHSM 監査ログ記録機能のリリース (2018 年 8 月 30 日) 前に初期化されたクラスターの CloudWatch Logs には表示されません。詳細については、「ドキュメント履歴」を参照してください。

次のイベント例には、クラスターの最初の HSM のログストリームを表示しています。ログの最初のイベント (Sequence No 0x0 がついているもの) は、HSM(CN_INIT_TOKEN) を初期化するコマンドを表しています。このレスポンスは、コマンドが正常に実行されたことを示します (Response : 0: HSM Return: SUCCESS)。

```
Time: 12/19/17 21:01:16.962174, usecs:1513717276962174
Sequence No : 0x0
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_TOKEN (0x1)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

このログストリーム例の 2 番目のイベント (Sequence No 0x1) は、HSM が使用するパスワード暗号化キーを作成するコマンド (CN_GEN_PSWD_ENC_KEY) を記録します。

これは、各クラスターの最初の HSM の一般的な起動シーケンスです。同じクラスターの後続の HSM は最初の HSM のクローンであるため、これらはおなじパスワード暗号化キーを使用します。

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

この例のログストリーミング (Sequence No 0x2) の 3 番目のイベントは、[Appliance User (AU)] が作成したもので、 AWS CloudHSM サービスです。HSM ユーザーに関係するイベントには、ユーザー名とユーザータイプ用の追加フィールドが含まれています。

```
Time: 12/19/17 21:01:17.174902, usecs:1513717277174902
Sequence No : 0x2
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_APPLIANCE_USER (0xfc)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
```

```
User Name : app_user
User Type : CN_APPLIANCE_USER (5)
```

このログストリーム例の 4 番目のイベント (Sequence No 0x3) は、HSM の初期化を完了する CN_INIT_DONE イベントを記録します。

```
Time: 12/19/17 21:01:17.298914, usecs:1513717277298914
Sequence No : 0x3
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_DONE (0x95)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

起動シーケンスの残りのイベントを追跡することができます。これらのイベントには、いくつかのログイン/ログアウトイベント、およびキー暗号化キー (KEK) の生成が含まれている場合があります。次のイベントは、Precrypto Officer (PRECO) のパスワードを変更するコマンドを記録します。このコマンドは、クラスターをアクティブ化します。

```
Time: 12/13/17 23:04:33.846554, usecs:1513206273846554
Sequence No: 0x1d
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_CHANGE_PSWD (0x9)
Session Handle: 0x2010003
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: admin
User Type: CN_CRYPTO_PRE_OFFICER (6)
```

ログインとログアウトイベント

監査ログを解釈するときに、ユーザーのロギングおよび HSM のログイン/ログアウトを記録するイベントに注目します。このイベントは、どのユーザーがログインとログアウト間のシーケンスに示される管理コマンドの責任者であるかを判断するために役立ちます。

たとえば、このログエントリは admin という名前の crypto officer のログインを記録しています。シーケンス番号 (0x0) は、これがこのログストリームの最初のイベントであることを示しています。

ユーザーが HSM にログインすると、このクラスターの他の HSM にもこのユーザーのログインイベントが記録されます。ログインイベントの開始からすぐに、クラスターの他の HSM のログストリームで該当するログインイベントが見つかります。

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

次のイベント例では、admin crypto officer のログアウトを記録しています。シーケンス番号 (0x2) は、これがログストリームの 3 番目のイベントであることを示しています。

ログインしているユーザーがログアウトせずにセッションを終了すると、ログストリームには、CN_LOGOUT イベントの代わりにCN_APP_FINALIZE あるいは終了セッションイベント (CN_SESSION_CLOSE) が含まれます。ログインイベントとは異なり、このログアウトイベントは通常の場合、このコマンドを実行する HSM にのみ記録されます。

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGOUT (0xe)
Session Handle : 0x7014000
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

ユーザー名が無効なためにログイン試行が失敗した場合、HSM はログインコマンドで提供されたユーザー名およびユーザータイプを CN_LOGIN イベントに記録します。このレスポンスには、ユーザー名が存在しないことを示すメッセージ 157 が表示されます。

```
Time: 01/24/18 17:41:39.037255, usecs:1516815699037255
Sequence No : 0x4
```

```
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 157:HSM Error: user isn't initialized or user with this name doesn't exist
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : ExampleUser
User Type : CN_CRYPTO_USER (1)
```

パスワードが無効なためにログイン試行が失敗した場合、HSM はログインコマンドで提供されたユーザー名およびユーザータイプを CN_LOGIN イベントに記録します。レスポンスには、RET_USER_LOGIN_FAILURE エラーコードを示すエラーメッセージが表示されます。

```
Time: 01/24/18 17:44:25.013218, usecs:1516815865013218
Sequence No : 0x5
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 163:HSM Error: RET_USER_LOGIN_FAILURE
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

例: ユーザーの作成と削除

この例には、crypto officer (CO) がユーザーの作成および削除をしたことを記録するログイベントが示されています。

最初のイベントでは、CO (admin) の HSM へのログインを記録しています。シーケンス番号 (0x0) は、これがログストリームの最初のイベントであることを示しています。このイベントには、ログインしたユーザーのユーザー名とタイプが含まれています。

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
```

```
User Type : CN_CRYPTO_OFFICER (2)
```

ログストリームの次のイベント (シーケンス 0x1) には、CO が新しい Crypto User (CU) を作成したことが記録されています。このイベントには、新しいユーザーのユーザー名とタイプが含まれています。

```
Time: 01/16/18 01:49:39.437708, usecs:1516067379437708
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_USER (0x3)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : bob
User Type : CN_CRYPTO_USER (1)
```

次に、CO は別の crypto officer (alice) を作成します。このシーケンス番号は、このアクションが前のアクションに従っていること (介在するアクションなしで) を示しています。

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_CO (0x4)
Session Handle : 0x7014007
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
User Type : CN_CRYPTO_OFFICER (2)
```

後で、admin という名前の CO がログインし、alice という名前の crypto officer を削除しています。HSM は CN_DELETE_USER イベントを記録します。このイベントには、削除されたユーザーのユーザー名とタイプが含まれています。

```
Time: 01/23/18 19:58:23.451420, usecs:1516737503451420
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_DELETE_USER (0xa1)
Session Handle : 0x7014007
```

```
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
User Type : CN_CRYPTO_OFFICER (2)
```

例: キーペアの作成と削除

この例では、キーペアを作成/削除したことを HSM 監査ログに記録したイベントを示しています。

次のイベントでは、crypto_user という名前の Crypto User (CU) が HSM にログインしたことを記録しています。

```
Time: 12/13/17 23:09:04.648952, usecs:1513206544648952
Sequence No: 0x28
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_LOGIN (0xd)
Session Handle: 0x2014005
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPTO_USER (1)
```

次に、CU がキーペア (CN_GENERATE_KEY_PAIR) を生成します。プライベートキーのキーハンドルは 131079 です。パブリックキーのキーハンドルは 131078 です。

```
Time: 12/13/17 23:09:04.761594, usecs:1513206544761594
Sequence No: 0x29
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_GENERATE_KEY_PAIR (0x19)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
Public Key Handle: 131078
```

CU はすぐにこのキーペアを削除します。CN_DESTROY_OBJECT イベントは、パブリックキー (131078) の削除を記録しています。

```
Time: 12/13/17 23:09:04.813977, usecs:1513206544813977
```

```
Sequence No: 0x2a
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131078
Public Key Handle: 0
```

次に、2 番目の CN_DESTROY_OBJECT イベントに、プライベートキー (131079) の削除が記録されています。

```
Time: 12/13/17 23:09:04.815530, usecs:1513206544815530
Sequence No: 0x2b
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
Public Key Handle: 0
```

最後に、CU がログアウトします。

```
Time: 12/13/17 23:09:04.817222, usecs:1513206544817222
Sequence No: 0x2c
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_LOGOUT (0xe)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPTO_USER (1)
```

例: キーの生成と同期

この例では、複数の HSM のクラスターでキーを作成した結果が示されています。1 つの HSM で生成されたキーはこの HSM からマスクされたオブジェクトとして抽出され、別の HSM にマスクされたオブジェクトとして挿入されます。

> **Note**
>
> クライアントツールで、キーを同期できない場合があります。または、特定数の HSM のみにキーを同期する min_srv パラメータがコマンドに含まれている場合があります。いずれの場合も、 AWS CloudHSM サービスはキーをクラスター内の他の HSMsに同期します。HSM はクライアント側の管理コマンドのみをログに記録するため、サーバー側の同期はこの HSM ログには記録されません。

まず、このコマンドを受信して実行する HSM のログストリームを検討します。このログストリームは HSM ID (hsm-abcde123456) で名付けられていますが、この HSM ID はログイベントには表示されません。

まず、 testuser Crypto User(CU) が hsm-abcde123456 HSM にログインします。

```
Time: 01/24/18 00:39:23.172777, usecs:1516754363172777
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

CU は exSymKey コマンドを実行して対称キーを生成します。hsm-abcde123456 HSM は 262152 のキーハンドルを使用して対称キーを生成します。HSM はそのログに CN_GENERATE_KEY イベントを記録します。

```
Time: 01/24/18 00:39:30.328334, usecs:1516754370328334
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GENERATE_KEY (0x17)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

hsm-abcde123456 のログストリームの次のイベントには、キーの同期プロセスの最初のステップが記録されています。新しいキー (キーハンドル 262152) は HSM からマスクオブジェクトとして抽出されています。

```
Time: 01/24/18 00:39:30.330956, usecs:1516754370330956
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

ここで、同じクラスターの別の HSM である HSM hsm-zyxwv987654 のログストリームを検討します。このログストリームにも、testuser CU のログインイベントが含まれています。時刻値は、ユーザーが hsm-abcde123456 HSM に ログインしたすぐあとで発生したことを示しています。

```
Time: 01/24/18 00:39:23.199740, usecs:1516754363199740
Sequence No : 0xd
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

HSM のこのログストリームには、CN_GENERATE_KEY イベントがありません。ただし、この HSM へのキーの同期を記録するイベントはあります。CN_INSERT_MASKED_OBJECT_USER イベントは、キー 262152 をマスクされたオブジェクトとして受信したことを記録しています。これで、キー 262152 がクラスターの両方の HSM に存在するようになりました。

```
Time: 01/24/18 00:39:30.408950, usecs:1516754370408950
Sequence No : 0xe
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0x7004003
```

```
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

CU ユーザーがログアウトすると、この CN_LOGOUT イベントはコマンドを受信する HSM のログストリームのみに表示されます。

例: キーのエクスポート

この例では、Crypto User (CU) が複数の HSM があるクラスターからキーをエクスポートすることを記録した監査ログイベントを示しています。

次のイベントは、CU (testuser) が [key_mgmt_util] にログインしたことを記録しています。

```
Time: 01/24/18 19:42:22.695884, usecs:1516822942695884
Sequence No : 0x26
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

CU は exSymKey コマンドを実行して、キー 7 (256 ビット AES キー) をエクスポートします。このコマンドは、ラップキーとして HSM でキー 6 (256 ビット AES キー) を使用します。

コマンドを受信した HSM は、エクスポートされたキー 7 の CN_WRAP_KEY イベントを記録します。

```
Time: 01/24/18 19:51:12.860123, usecs:1516823472860123
Sequence No : 0x27
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_WRAP_KEY (0x1a)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 7
Public Key Handle : 0
```

次に、HSM はラップされたキーであるキー 6 の CN_NIST_AES_WRAP イベントを記録します。このキーはラップされ、すぐにラップ解除されますが、HSM は 1 つのイベントのみを記録します。

```
Time: 01/24/18 19:51:12.905257, usecs:1516823472905257
Sequence No : 0x28
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
Public Key Handle : 0
```

この exSymKey コマンドはエクスポートされたキーをファイルに書き込みますが、HSM でキーの変更は行いません。したがって、クラスターの他の HSM のログには対応するイベントはありません。

例: キーのインポート

この例では、クラスターの HSM にキーをインポートしたことを記録する監査ログイベントを示しています。この例では、Crypto User (CU) が imSymKey コマンドを使用して AES キーを HSM にインポートしています。このコマンドは、キー 6 をラップされたキーとして使用します。

コマンドを受信した HSM は、ラップされたキー 6 の CN_NIST_AES_WRAP イベントをまず記録します。

```
Time: 01/24/18 19:58:23.170518, usecs:1516823903170518
Sequence No : 0x29
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
Public Key Handle : 0
```

次に、この HSM はインポートオペレーションを表す CN_UNWRAP_KEY イベントを記録します。インポートされたキーには、11 のキーハンドルが割り当てられます。

```
Time: 01/24/18 19:58:23.200711, usecs:1516823903200711
Sequence No : 0x2a
```

```
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_UNWRAP_KEY (0x1b)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

新しいキーが生成あるいはインポートされると、クライアントツールは自動的にこの新しいキーをクラスターの他の HSM に同期する試みを行います。この場合、HSM はキー 11 がマスクオブジェクトとして HSM から抽出されたことを CN_EXTRACT_MASKED_OBJECT_USER イベントに記録します。

```
Time: 01/24/18 19:58:23.203350, usecs:1516823903203350
Sequence No : 0x2b
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

クラスターの他の HSM のログストリームには、新しくインポートされたキーの到着が示されます。

たとえば、同じクラスターの異なる HSM のログストリームには、このイベントが記録されています。この CN_INSERT_MASKED_OBJECT_USER イベントは、キー 11 を表すマスクされたオブジェクトの到着を記録します。

```
Time: 01/24/18 19:58:23.286793, usecs:1516823903286793
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

例: キーの共有と共有解除

この例では、Crypto User (CU) が他の Crypto User (CU) とECC プライベートキーを共有または共有解除したときに記録される監査ログイベントを示します。CU は、shareKey コマンドを使用し、キーのハンドル、ユーザー ID、値 1 (共有) または 0 (共有解除) を行います。

次の例では、コマンドを受け取る HSM は、共有オペレーションを表す CM_SHARE_OBJECT イベントを記録します。

```
Time: 02/08/19 19:35:39.480168, usecs:1549654539480168
Sequence No : 0x3f
Reboot counter : 0x38
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_SHARE_OBJECT (0x12)
Session Handle : 0x3014007
Response : 0:HSM Return: SUCCESS
Log type : UNKNOWN_LOG_TYPE (5)
```

AWS CloudHSM 監査ログリファレンス

AWS CloudHSM は、監査ログイベントに HSM 管理コマンドを記録します。各イベントには、発生したアクションとそのレスポンスを識別するオペレーションコード (0pcode) 値があります。0pcode 値を使用して、ログの検索、ソート、フィルタリングができます。

次の表は、 AWS CloudHSM 監査ログ0pcodeの値を定義します。

オペレーションコード (Opcode)	説明
[ユーザーログイン]: これらのイベントにはユーザー名とユーザータイプが含まれます	
CN_LOGIN (0xd)	ユーザーログイン
CN_LOGOUT (0xe)	ユーザーログアウト
CN_APP_FINALIZE	HSM との接続が閉じられました。この接続からセッションキーまたはクォーラムトークンが削除されました。
CN_CLOSE_SESSION	HSM とのセッションが終了しました。このセッションのセッションキーまたはクォーラムトークンはすべて削除されました。

オペレーションコード (Opcode)	説明
ユーザー管理: これらのイベントにはユーザー名とユーザータイプが含まれます	
CN_CREATE_USER (0x3)	Crypto User (CU) を作成する
CN_CREATE_CO	Crypto Officer (CO) を作成する
CN_DELETE_USER	ユーザーの削除
CN_CHANGE_PSWD	ユーザーのパスワードを変更する
CN_SET_M_VALUE	Set クォーラム認証 (M of N) for a user action
CN_APPROVE_TOKEN	Approve a クォーラム認証 token for a user action
CN_DELETE_TOKEN	Delete one or more クォーラムトークン
CN_GET_TOKEN	Request a signing token to initiate a クォーラムオペレーション
キー管理: このイベントにはキーハンドルが含まれます	
CN_GENERATE_KEY	対称キーの生成
CN_GENERATE_KEY_PAIR (0x19)	Generate an asymmetric key pair
CN_CREATE_OBJECT	Import a public key (without wrapping)
CN_MODIFY_OBJECT	Set a key attribute
CN_DESTROY_OBJECT (0x11)	Deletion of a セッションキー
CN_TOMBSTONE_OBJECT	Deletion of a トークンキー
CN_SHARE_OBJECT	キーの共有あるいは非共有
CN_WRAP_KEY	Export an encrypted copy of a key (wrapKey)
CN_UNWRAP_KEY	Import an encrypted copy of a key (UnwrapKey)

オペレーションコード (Opcode)	説明
CN_DERIVE_KEY	Derive a symmetric key from an existing key
CN_NIST_AES_WRAP	AES キーを使用してキーを暗号化または複合化
CN_INSERT_MASKED_OBJECT_USER	Insert an encrypted key with attributes from another HSM in the cluster.
CN_EXTRACT_MASKED_OBJECT_USER	Wraps/encrypts a key with attributes from the HSM to be sent to another HSM in the cluster.
Session Management	
CN_ENCRYPT_SESSION_V2 (0x107)	Establishes an authenticated end-to-end encrypted session.
END_MARKER_OPCODE (0xffff)	Inserts an end-marker in the audit logs buffer indicating no more loggable commands are allowed on the HSM
Back up HSMs	
CN_BACKUP_BEGIN	Begin the backup process
CN_BACKUP_END	Completed the backup process
CN_RESTORE_BEGIN	Begin restoring from a backup
CN_RESTORE_END	Completed the restoration process from a backup
Certificate-Based Authentication	
CN_CERT_AUTH_STORE_CERT	Stores the cluster certificate
HSM Instance Commands	
CN_INIT_TOKEN (0x1)	Start the HSM initialization process

オペレーションコード (Opcode)	説明
`CN_INIT_DONE`	The HSM initialization process has finished
`CN_GEN_KEY_ENC_KEY`	Generate a key encryption key (KEK)
`CN_GEN_PSWD_ENC_KEY (0x1d)`	Generate a password encryption key (PEK)
HSM crypto commands	
`CN_FIPS_RAND`	Generate a FIPS-compliant random number[1]

[1] hsm1.medium クラスターに対してのみログに記録されます。

の CloudWatch メトリクスの取得 AWS CloudHSM

CloudWatch を使用して AWS CloudHSM クラスターをリアルタイムでモニタリングします。メトリクスは、リージョン、クラスター ID、またはクラスター ID と HSM ID ごとにグループ化できます。

`AWS/CloudHSM` 名前空間には、次のメトリクスが含まれます。

メトリクス	説明
HsmUnhealthy	HSM インスタンスが正常に動作していません。 は、異常なインスタンス AWS CloudHSM を自動的に置き換えます。HSM の置き換え中は、パフォーマンスへの影響を抑えるために、クラスターサイズを積極的に拡大することもできます。
HsmTemperature [1]	ハードウェアプロセッサのジャンクション温度です。温度が 110 度に達すると、システムがシャットダウンします。
HsmKeysSessionOccupied	HSM インスタンスにより使用されているセッションキーの数です。
HsmKeysTokenOccupied	HSM インスタンスとクラスターにより使用されるトークンキーの数です。

メトリクス	説明
HsmSslCtxsOccupied [1]	HSM インスタンスで現在確立されているエンドツーエンドの暗号化チャネルの数です。最大 2,048 のチャネルが許容されます。
HsmSessionCount	HSM インスタンスへのオープン接続の数です。最大 2,048 の接続が許容されます。デフォルトでは、クライアントデーモンは、1 つのend-to-end暗号化されたチャネルで、各 HSM インスタンスで 2 つのセッションを開くように設定されています。また、HSM の正常性をモニタリングするために、HSM で最大 2 つの接続が開いている AWS CloudHSM 場合があります HSMs 。
HsmUsersAvailable	作成可能な追加ユーザーの数です。これは、ユーザーの最大数 (HsmUsersMax に表示) から現在までに作成されたユーザー数を引いた数に相当します。
HsmUsersMax [1]	HSM インスタンスで作成可能なユーザーの最大数です。
InterfaceEth2OctetsInput [1]	現在の HSM への受信トラフィックの累積合計です。
InterfaceEth2OctetsOutput [1]	現在の HSM への送信トラフィックの累積合計です。

- [1] このメトリクスは hsm2m.medium では使用できません。

AWS CloudHSM パフォーマンス情報

本番 AWS CloudHSM クラスターの場合、リージョン内の異なるアベイラビリティーゾーンに少なくとも 2 つのハードウェアセキュリティモジュール (HSM) インスタンスが分散されている必要があります。クラスターの負荷テストを行って予測すべきピーク負荷を決定し、高可用性を確保するためにクラスターに HSM を 1 つ追加することを推奨します。新しく生成されるキーの耐久性を必要とするアプリケーションの場合は、リージョン内の異なるアベイラビリティーゾーンに 3 つの HSM インスタンスを分散させることをお勧めします。

パフォーマンスデータ

AWS CloudHSM クラスターのパフォーマンスは、特定のワークロードによって異なります。パフォーマンスを向上させるために、クラスターに HSM インスタンスを追加できます。パフォーマンスは、設定、データサイズ、および EC2 インスタンスにかかる追加のアプリケーション負荷によって異なる場合があります。アプリケーションの負荷テストを行い、スケーリングの必要性を判断することをお勧めします。

次の表は、hsm1.medium インスタンスを備えた EC2 インスタンスで実行される一般的な暗号化アルゴリズムのおおよそのパフォーマンスを示しています。

hsm1.medium のパフォーマンスデータ

運用	2 つの HSM クラスター [1]	3 つの HSM クラスター [2]	6 つの HSM クラスター [3]
RSA 2,048 ビットサイン	2,000 オペレーション/秒	3,000 オペレーション/秒	5,000 オペレーション/秒
EC P256 サイン	500 オペレーション/秒	750 オペレーション/秒	1,500 オペレーション/秒

次の表は、hsm2m.medium を使用して EC2 インスタンスで実行される一般的な暗号化アルゴリズムのおおよそのパフォーマンスを示しています。

hsm2m.medium のパフォーマンスデータ

運用	2 つの HSM クラスター [1]	3 つの HSM クラスター [2]	6 つの HSM クラスター [3]
RSA 2,048 ビットサイン	2,000 オペレーション/秒	3,000 オペレーション/秒	5,000 オペレーション/秒
EC P256 サイン	3,000 オペレーション/秒	4,500 オペレーション/秒	7,000 オペレーション/秒
ML-DSA-44 署名	80 オペレーション/秒	140 オペレーション/秒	190 オペレーション/秒

- [1] 1 つの c4.large EC2 インスタンスと EC2 インスタンスと同じ AZ にある 1 つの HSM で実行されている Java マルチスレッドアプリケーションを備えた 2 つの HSM クラスター。
- [2] 1 つの c4.large EC2 インスタンスと EC2 インスタンスと同じ AZ にある 1 つの HSM で実行されている Java マルチスレッドアプリケーションを備えた 3 つの HSM クラスター。
- [3] 1 つの c4.large EC2 インスタンスと EC2 インスタンスと同じ AZ にある 1 つの HSM で実行されている Java マルチスレッドアプリケーションを備えた 6 つの HSM クラスター。

HSM スロットリング

ワークロードがクラスターの HSM 容量を超えると、HSM がビジー状態またはスロットリングされていることを示すエラーメッセージが表示されます。この場合の対処方法の詳細については、HSM スロットリング を参照してください

のセキュリティ AWS CloudHSM

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からメリットを得られます。

セキュリティは、 AWS 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とクラウド内のセキュリティとして説明しています:

- クラウドのセキュリティ – AWS は、 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責任を担います AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。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。が適用されるコンプライアンスプログラムの詳細については AWS CloudHSM、「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。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、 を使用する際の責任共有モデルの適用方法を理解するのに役立ちます AWS CloudHSM。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成する AWS CloudHSM ように を設定する方法について説明します。また、 AWS CloudHSM リソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

内容

- IAM ポリシーによる API アクセスコントロール
- でのデータ保護 AWS CloudHSM
- の ID とアクセスの管理 AWS CloudHSM
- のコンプライアンス検証 AWS CloudHSM
- の耐障害性 AWS CloudHSM
- のインフラストラクチャセキュリティ AWS CloudHSM
- AWS CloudHSM および VPC エンドポイント
- で管理を更新する AWS CloudHSM

IAM ポリシーによる API アクセスコントロール

IAM ポリシーを IPv6 にアップグレードする

AWS CloudHSM のお客様は、IAM ポリシーを使用して AWS CloudHSM APIs、設定された範囲外の IP アドレスが AWS CloudHSM APIs にアクセスできないようにします。

AWS CloudHSM APIs がホストされている cloudhsmv2.*<region>*.api.aws デュアルスタックエンドポイントは、IPv4 に加えて IPv6 をサポートします。 IPv4

IPv4 と IPv6 の両方をサポートする必要があるお客様は、IPv6 アドレスを処理するように IP アドレスフィルタリングポリシーを更新する必要があります。更新しないと、IPv6 AWS CloudHSM 経由でに接続する機能に影響します。

アップグレードすべきるユーザーとは?

aws:sourceIp を含むポリシーでデュアルアドレスを使用しているお客様は、このアップグレードの影響を受けます。デュアルアドレス指定とは、ネットワークが IPv4 と IPv6 の両方をサポートすることを意味します。

デュアルアドレス指定を使用している場合は、現在 IPv4 形式のアドレスで構成されている IAM ポリシーを、IPv6 形式のアドレスを含むように更新する必要があります。

アクセスに関する問題については、サポート にお問い合わせください。

> **Note**
>
> 次のお客様は、アップグレードの影響は受けません。
>
> - IPv4 ネットワークのみを利用しているお客様。

IPv6 とは

IPv6 は、最終的に IPv4 を IPv6 に置き換えることを意図した次世代の IP 規格です。以前のバージョンの IPv4 は、32 ビットのアドレス指定方式を使用して 43 億台のデバイスをサポートしていました。IPv6 は代わりに 128 ビットのアドレス指定を使用して、約 340 兆 x1 兆倍 x1 兆倍 (つまり 2 の 128 乗) のデバイスをサポートします。

詳細については、VPC IPv6 ウェブページ を参照してください。

```
2001:cdba:0000:0000:0000:0000:3257:9652
2001:cdba:0:0:0:3257:9652
2001:cdba::3257:965
```

IPv6 用の IAM ポリシーを更新する

現在、IAM ポリシーは、aws:SourceIp フィルターを使用して IP アドレスの許容範囲を設定するために使用されています。

デュアルアドレス指定では、IPv4 と IPv6 の両方のトラフィックをサポートしています。ネットワークでデュアルアドレス指定を使用している場合、IPv6 アドレス範囲を含めるように、IP アドレスフィルタリングに使用される IAM ポリシーを更新する必要があります。

例えば、以下のポリシーは、Condition 要素で許可された IPv4 アドレス範囲の192.0.2.0.* と 203.0.113.0.* を識別します。

```
# https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_policies_examples_aws_deny-ip.html
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "*aws:SourceIp*": [
                    "*192.0.2.0/24*",
                    "*203.0.113.0/24*"
                ]
            },
            "Bool": {
                "aws:ViaAWSService": "false"
            }
        }
    }
}
```

このポリシーを更新するには、Condition 要素を変更して、IPv6 アドレス範囲 の 2001:DB8:1234:5678::/64 と 2001:cdba:3257:8593::/64 を含むように更新します。

> **Note**
>
> 既存の IPv4 アドレスは下位互換性のために必要なため、削除しないでください。

```
"Condition": {
                "NotIpAddress": {
                    "*aws:SourceIp*": [
                        "*192.0.2.0/24*", <<DO NOT REMOVE existing IPv4 address>>
                        "*203.0.113.0/24*", <<DO NOT REMOVE existing IPv4 address>>
                        "*2001:DB8:1234:5678::/64*", <<New IPv6 IP address>>
                        "*2001:cdba:3257:8593::/64*" <<New IPv6 IP address>>
                    ]
                },
                "Bool": {
                    "aws:ViaAWSService": "false"
                }
            }
```

クライアントが IPv6 をサポートしていることを検証する

cloudhsmv2.{region}.api.aws エンドポイントを使用しているお客様は、接続できるかどうかを確認することをお勧めします。以下の手順では、検証を実行する方法について説明します。

この例では、Linux および curl バージョン 8.6.0 を使用し、api.aws AWS CloudHSM エンドポイントにある IPv6 対応エンドポイントを持つサービスエンドポイントを使用します。 IPv6

> **Note**
>
> を、クライアントが配置されているのと同じリージョン AWS リージョン に切り替えます。この例では、米国東部 (バージニア北部) – us-east-1 エンドポイントを使用します。

1. 次の dig コマンドを使用して、エンドポイントが IPv6 アドレスで解決されることを確認します。

```
dig +short AAAA cloudhsmv2.us-east-1.api.aws
2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```

2. 次の curl コマンドを使用して、クライアント ネットワークが IPv6 接続を確立できるかどうかを確認します。レスポンスコードが 404 の場合は接続が成功したことを意味し、0 の場合は接続が失敗したことを意味します。

```
curl --ipv6 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://cloudhsmv2.us-east-1.api.aws

remote ip: 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
response code: 404
```

リモート IP アドレスが特定され、さらにレスポンスコードが 0 でない場合は、IPv6 を使用してエンドポイントへのネットワーク接続が正常に確立されています。リモート IP は IPv6 アドレスである必要があります。これは、オペレーティングシステムがクライアントに有効なプロトコルを選択する必要があるからです。リモート IP が IPv6 アドレスでない場合は、次のコマンドを使用して、強制的に curl が IPv4 を使用するようにします。

```
curl --ipv4 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://cloudhsmv2.us-east-1.api.aws

remote ip: 3.123.154.250
response code: 404
```

リモート IP が空白の場合、またはレスポンスコードが 0 の場合は、クライアントネットワークまたはエンドポイントへのネットワークパスは IPv4 専用です。これを確認するには、次の curl コマンドを使用します。

```
curl -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
 %{response_code}\n" https://cloudhsmv2.us-east-1.api.aws

remote ip: 3.123.154.250
response code: 404
```

でのデータ保護 AWS CloudHSM

AWS 責任共有モデル は、 AWS CloudHSMでのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と

管理タスクもユーザーの責任となります。データプライバシーの詳細については、「Data Privacy FAQChina」を参照してください。 欧州におけるデータ保護に関する情報については、General Data Protection Regulation (GDPR) Center を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール AWS CloudHSM 、API、または SDK を使用して AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

保管中の暗号化

AWS CloudHSM が HSM からバックアップを作成すると、HSM は送信前にデータを暗号化します AWS CloudHSM。データは、一意の一時的な暗号化キーを使用して暗号化されます。詳細については、「AWS CloudHSM クラスターのバックアップ」を参照してください。

転送中の暗号化

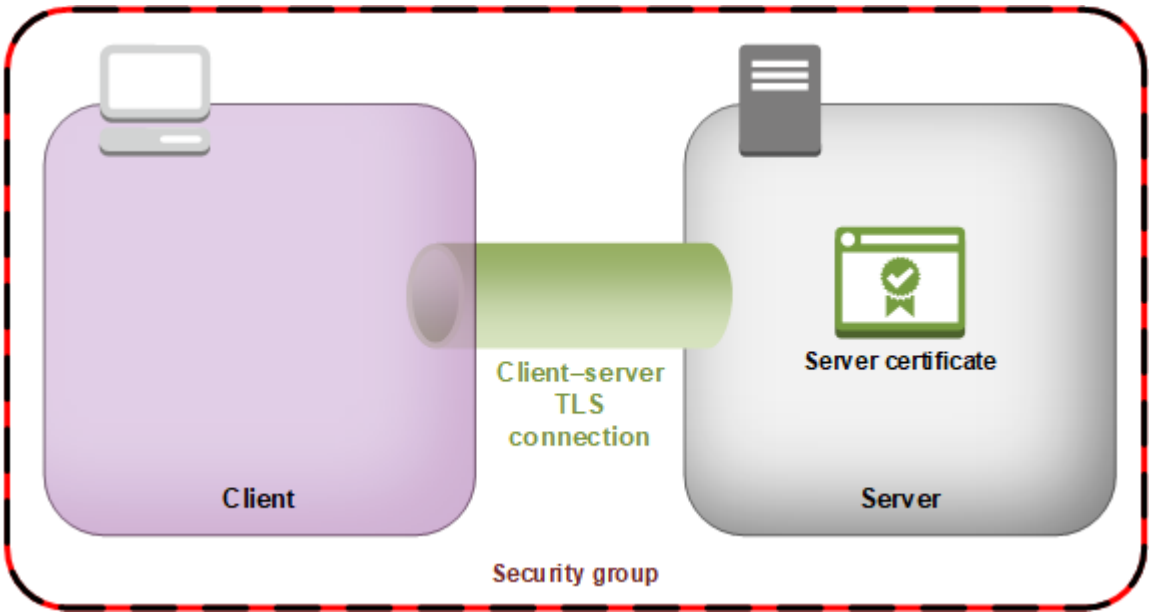
AWS CloudHSM クライアントとクラスター内の HSM 間の通信は、エンドツーエンドで暗号化されます。この通信は、クライアントと HSM によってのみ復号できます。詳細については、「エンドツーエンドの暗号化」を参照してください。

AWS CloudHSM クライアントend-to-end暗号化

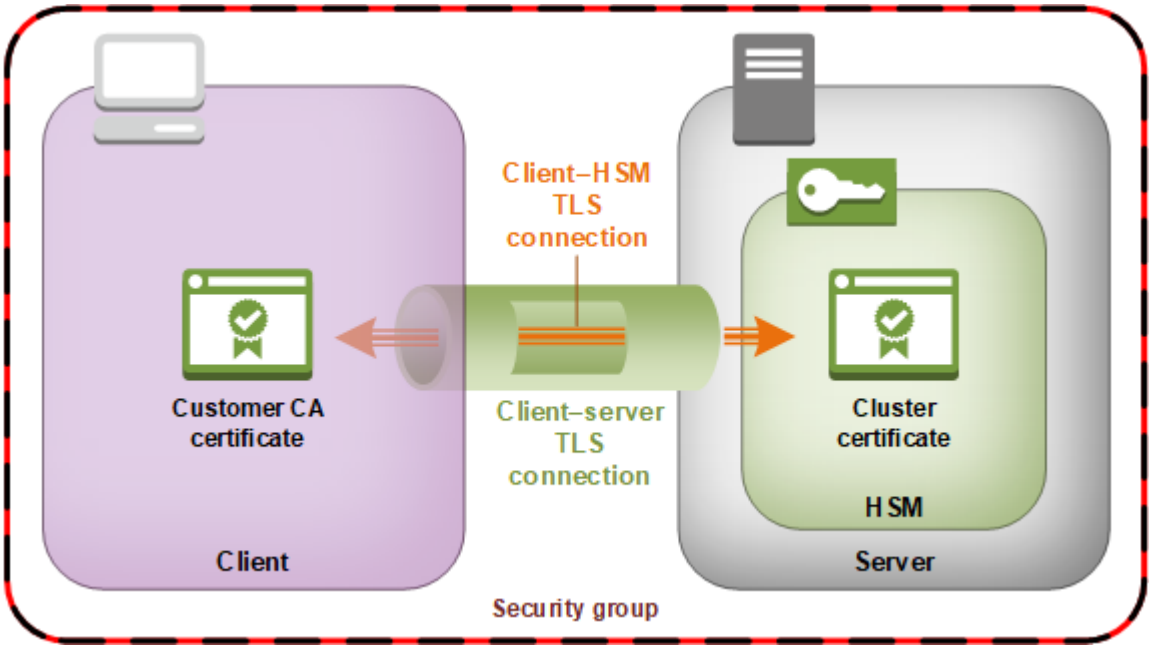
クライアントインスタンスとクラスターの HSM の間の通信はエンドツーエンドで暗号化されます。クライアントと HSM のみが、通信を復号できます。

次のプロセスでは、クライアントが HSM とのエンドツーエンドの暗号化通信を確立する方法を説明します。

1. クライアントは、HSM ハードウェアをホストするサーバーと Transport Layer Security (TLS) 接続を確立します。クラスターのセキュリティグループによって、セキュリティグループのクライアントインスタンスからのみ、サーバーへのインバウンドトラフィックが許可されます。また、クライアントはサーバーの証明書をチェックして、それが信頼されたサーバーであることを確認します。



2. 次に、クライアントは HSM ハードウェアと暗号化された接続を確立します。HSM には、独自の認証局 (CA) で署名されたクラスター証明書があり、クライアントには CA のルート証明書があります。クライアントと HSM の暗号化された接続が確立される前に、クライアントはルート証明書に対して HSM のクラスター証明書を確認します。接続が確立されるのは、HSM が信頼済みであることをクライアントが正しく確認した場合のみです。



クラスターバックアップのセキュリティ

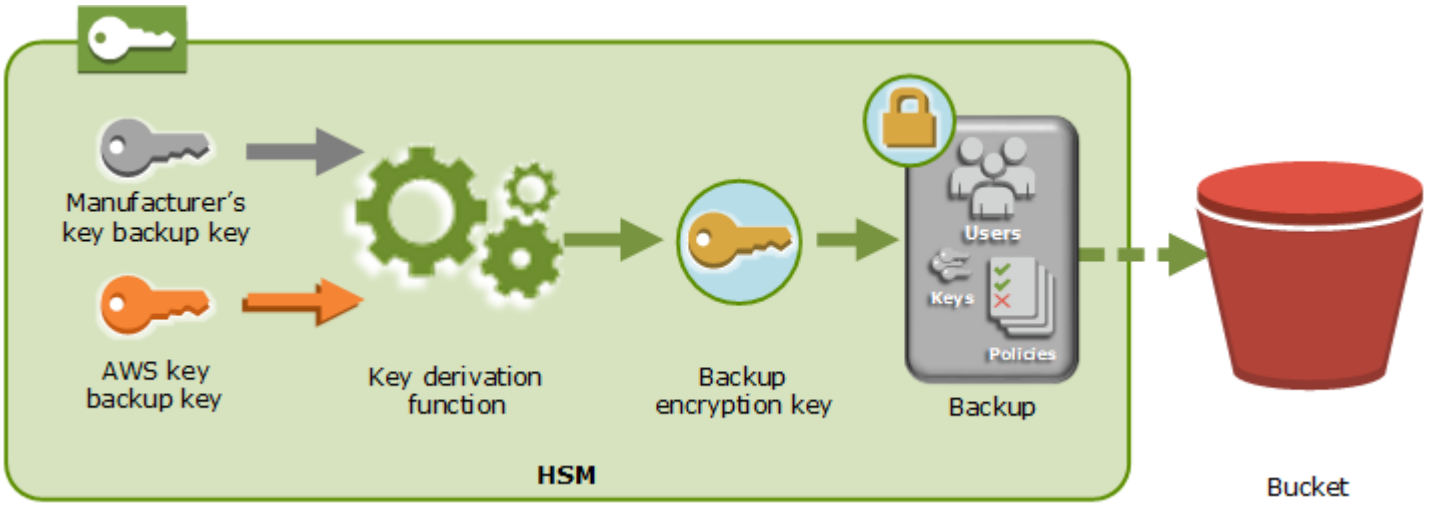
が HSM からバックアップ AWS CloudHSM を作成すると、HSM は送信前にすべてのデータを暗号化します AWS CloudHSM。データがプレーンテキスト形式で HSM から外部に出ることはありません。さらに、 はバックアップの復号に使用されるキーにアクセス AWS できない AWS ため、 によるバックアップの復号化はできません。

データを暗号化するために、HSM はエフェメラルバックアップキー (EBK) として知られる一意の暗号化キーを一時的に使用します。EBK は、 がバックアップ AWS CloudHSM を作成するときに HSM 内で生成される AES 256 ビット暗号化キーです。HSM は EBK を生成し、それを使用して NIST special publication 800-38F に準拠する FIPS 承認 AES キーラップメソッドで HSM のデータを暗号化します。次に、HSM は暗号化されたデータを に渡します AWS CloudHSM。暗号化されたデータには、EBK の暗号化済みコピーが含まれています。

EBK を暗号化するために、HSM は永続的バックアップキー (PBK) として知られる別の暗号化キーを使用します。PBK も、AES 256 ビット暗号化キーです。PBK を生成するために、HSM は NIST special publication 800-108 に準拠する FIPS 承認キー取得機能 (KDF) をカウンターモードで使用します。この KDF への入力には次のものがあります。

- ハードウェア製造元が HSM に永続的に埋め込んだ、製造元キーバックアップキー (MKBK)。
- AWS キーバックアップキー (AKBK)。最初の設定時に HSM に安全にインストールされます AWS CloudHSM。

次の図に暗号化プロセスがまとめてあります。バックアップ暗号化キーは、永続的なバックアップキー (PBK) とエフェメラルバックアップキー (EBK) を指します。



AWS CloudHSM は、同じ製造元によって作成された所有 AWSの HSMs にのみバックアップを復元できます。すべてのバックアップがすべてのユーザー、キー、およびオリジナルの HSM を含んでいるため、復元された HSM はオリジナルと同じ保護およびアクセス制御を含んでいます。復元されたデータは、復元前に HSM にあった可能性がある他のデータをすべて上書きします。

バックアップは暗号化されたデータのみで構成されます。サービスが Amazon S3 にバックアップを保存する前に、サービスは AWS Key Management Service () を使用してバックアップを再度暗号化しますAWS KMS。

の ID とアクセスの管理 AWS CloudHSM

AWS ではセキュリティ認証情報を使用して、ユーザーを識別し、AWS リソースへのアクセスを付与します。 AWS Identity and Access Management (IAM) の機能を使用すると、他のユーザー、サービス、アプリケーションが AWS リソースを完全または制限付きで使用できます。その際、お客様のセキュリティ認証情報は共有されません。

デフォルトでは、IAM ユーザーには、AWS リソースを作成、表示、変更するためのアクセス権限はありません。ロードバランサーなどのリソースにアクセスすること、およびタスクを実行することを IAM ユーザーに許可するには、次の操作を行います。

1. 必要な特定のリソースと API アクションを使用するアクセス許可を IAM ユーザーに付与する IAM ポリシーを作成します。
2. IAM ユーザーまたは IAM ユーザーが属するグループに、ポリシーをアタッチします。

ポリシーをユーザーまたはユーザーのグループにアタッチする場合、ポリシーによって特定リソースの特定タスクを実行するユーザーの権限が許可または拒否されます。

たとえば、IAM を使用して、お客様の AWS アカウントでユーザーとグループを作成できます。IAM ユーザーは、人、システム、またはアプリケーションです。その後、ユーザーとグループにアクセス許可を付与すると、IAM ポリシーを使用して指定したリソースに対する特定のアクションを実行できます。

IAM ポリシーを使用したアクセス権限の付与

ポリシーをユーザーまたはユーザーのグループにアタッチする場合、ポリシーによって特定リソースの特定タスクを実行するユーザーの権限が許可または拒否されます。

IAM ポリシーは、1 つ、または複数のステートメントで構成される JSON ドキュメントです。

次の IAM ポリシーの例では、米国東部 (バージニア北部) で AWS CloudHSM バックアップを記述することをユーザーに許可します。米国東部 (バージニア北部) からのリクエストのみを記述できます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudhsm:DescribeBackups",
      "Resource": "arn:aws:cloudhsm:us-east-1:123456789012:backup/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-1"
         }
      }
    }
  ]
}
```

- [Effect (効果)] — effect は、`Allow` または `Deny` にすることができます。デフォルトでは、IAM ユーザーはリソースおよび API アクションを使用するアクセス許可がないため、リクエストはす

べて拒否されます。明示的な許可はデフォルトに優先します。明示的な拒否はすべての許可に優先します。

- Action (アクション) — action は、アクセス許可を付与または拒否する対象とする、特定の API アクションです。アクション条件を指定する方法については、の API アクション AWS CloudHSM を参照してください。
- リソース — action. AWS CloudHSM does の影響を受けるリソースは、リソースレベルのアクセス許可をサポートしていません。すべての AWS CloudHSM リソースを指定するには、* ワイルドカードを使用する必要があります。
- [Condition (条件)] — ポリシーが有効になるタイミングを制御する条件を必要に応じて使用できます。詳細については、「の条件キー AWS CloudHSM」を参照してください。

詳細については、IAM ユーザーガイドを参照してください。

の API アクション AWS CloudHSM

IAM ポリシーステートメントの Action 要素で、 AWS CloudHSM が提供する任意の API アクションを指定できます。次の例に示すように、アクション名の前に小文字の文字列 `cloudhsm:` を指定する必要があります。

```
"Action": "cloudhsm:DescribeClusters"
```

1 つのステートメントで複数のアクションを指定するには、次の例に示すように、アクションをカンマで区切って全体を角括弧で囲みます。

```
"Action": [
    "cloudhsm:DescribeClusters",
    "cloudhsm:DescribeHsm"
]
```

ワイルドカード (*) を使用して複数のアクションを指定することもできます。次の例では、 で始 AWS CloudHSM まる のすべての API アクション名を指定しますList。

```
"Action": "cloudhsm:List*"
```

のすべての API アクションを指定するには AWS CloudHSM、次の例に示すように * ワイルドカードを使用します。

```
"Action": "cloudhsm:*"
```

の API アクションのリストについては AWS CloudHSM、AWS CloudHSM 「アクション」を参照してください。

の条件キー AWS CloudHSM

ポリシーを作成するときは、ポリシーをいつ有効にするか制御する条件を指定できます。各条件には 1 つ以上のキーと値のペアが含まれます。グローバル条件キーとサービス固有の条件キーがあります。

AWS CloudHSM には、サービス固有のコンテキストキーはありません。

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 global condition context keys を参照してください。

の事前定義された AWS 管理ポリシー AWS CloudHSM

AWS によって作成された管理ポリシーは、一般的ユースケースに必要なアクセス権限を付与します。これらのポリシーを、 AWS CloudHSM に対して必要なアクセス権に基づいて IAM ユーザーにアタッチできます。

- AWSCloudHSMFullAccess — AWS CloudHSM 機能を使用するために必要なフルアクセスを付与します。
- AWSCloudHSMReadOnlyAccess — AWS CloudHSM 機能への読み取り専用アクセスを許可します。

のカスタマー管理ポリシー AWS CloudHSM

の実行に必要なアクセス許可のみ AWS CloudHSM を含む の IAM 管理者グループを作成することをお勧めします AWS CloudHSM。適切なアクセス許可を持つポリシーをこのグループにアタッチします。必要に応じて、IAM ユーザーをグループに追加します。追加する各ユーザーは、管理者グループからポリシーを継承します。

また、ユーザーが必要とする権限に基づいて、追加のユーザーグループを作成することをお勧めします。これにより、信頼されたユーザーのみが重要な API アクションにアクセスできるようになります。たとえば、クラスターと HSM への読み取り専用アクセスを許可するために使用するユーザー

グループを作成できます。このグループでは、ユーザーがクラスターまたは HSM を削除できないため、信頼できないユーザーが運用ワークロードの可用性に影響を与えることがありません。

新しい AWS CloudHSM 管理機能が時間の経過とともに追加されるため、信頼できるユーザーのみがすぐにアクセスできるようになります。作成時に、制限付きのアクセス許可をポリシーに割り当てることで、後に新しい機能のアクセス許可を手動でユーザーに割り当てることができます。

のポリシーの例を次に示します AWS CloudHSM。ポリシーの作成方法とIAM ユーザーグループへのアタッチ方法の詳細については、IAM ユーザーガイド の 「[JSON] タブでのポリシーの作成 」を参照してください。

例

- 読み取り専用のアクセス許可
- パワーユーザーのアクセス許可
- 管理者権限

Example例: 読み取り専用アクセス許可

このポリシーでは、`DescribeClusters` および `DescribeBackups` API アクションへのアクセスを許可します。また、このポリシーには、特定の Amazon EC2 API アクションのアクセス許可が含まれています。このポリシーでは、ユーザーはクラスターまたは HSM を削除することはできません。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:ListTags"
      ],
      "Resource": "*"
   }
}
```

Example例: パワーユーザーのアクセス許可

このポリシーは、 AWS CloudHSM API アクションのサブセットへのアクセスを許可します。また、このポリシーには特定の Amazon EC2 アクションのアクセス許可が含まれています。このポリシーでは、ユーザーはクラスターまたは HSM を削除することはできません。アカウント AWS CloudHSM で が AWSServiceRoleForCloudHSM サービスにリンクされたロールを自動的に作成できるようにするには、 `iam:CreateServiceLinkedRole`アクションを含める必要があります。このロールにより、 はイベント AWS CloudHSM をログに記録できます。詳細については、「のサービスにリンクされたロール AWS CloudHSM」を参照してください。

> **Note**
>
> 各 API の特定の権限を確認するには、「サービス認可リファレンス AWS CloudHSM」の「のアクション、リソース、および条件キー」を参照してください。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:CreateCluster",
         "cloudhsm:CreateHsm",
         "cloudhsm:RestoreBackup",
         "cloudhsm:CopyBackupToRegion",
         "cloudhsm:InitializeCluster",
         "cloudhsm:ListTags",
         "cloudhsm:TagResource",
         "cloudhsm:UntagResource",
         "ec2:CreateNetworkInterface",
         "ec2:DescribeNetworkInterfaces",
         "ec2:DescribeNetworkInterfaceAttribute",
         "ec2:DetachNetworkInterface",
         "ec2:DeleteNetworkInterface",
         "ec2:CreateSecurityGroup",
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:AuthorizeSecurityGroupEgress",
```

```
            "ec2:RevokeSecurityGroupEgress",
            "ec2:DescribeSecurityGroups",
            "ec2:DeleteSecurityGroup",
            "ec2:CreateTags",
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "*"
    }
}
```

Example例: 管理者権限

このポリシーは、HSMs とクラスターを削除するアクションを含む、すべての AWS CloudHSM API アクションへのアクセスを許可します。また、このポリシーには特定の Amazon EC2 アクションのアクセス許可が含まれています。アカウント AWS CloudHSM で が AWSServiceRoleForCloudHSM サービスにリンクされたロールを自動的に作成できるようにするには、 `iam:CreateServiceLinkedRole`アクションを含める必要があります。このロールにより、はイベント AWS CloudHSM をログに記録できます。詳細については、「のサービスにリンクされたロール AWS CloudHSM」を参照してください。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":[
         "cloudhsm:*",
         "ec2:CreateNetworkInterface",
         "ec2:DescribeNetworkInterfaces",
         "ec2:DescribeNetworkInterfaceAttribute",
         "ec2:DetachNetworkInterface",
         "ec2:DeleteNetworkInterface",
         "ec2:CreateSecurityGroup",
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:AuthorizeSecurityGroupEgress",
         "ec2:RevokeSecurityGroupEgress",
         "ec2:DescribeSecurityGroups",
         "ec2:DeleteSecurityGroup",
```

```
            "ec2:CreateTags",
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "iam:CreateServiceLinkedRole"
        ],
        "Resource":"*"
    }
}
```

のサービスにリンクされたロール AWS CloudHSM

以前に に作成した IAM ポリシーには、`iam:CreateServiceLinkedRole`action. AWS CloudHSM defines a service-linked role named AWSServiceRoleForCloudHSM のカスタマー管理ポリシー AWS CloudHSMが含まれています。ロールは によって事前定義 AWS CloudHSM されており、ユーザーに代わって他の AWS サービスを呼び出す AWS CloudHSM ために必要なアクセス許可が含まれています。ロールは、ロールポリシーと信頼ポリシーのアクセス許可を手動で追加する必要がないため、サービスを簡単に設定できます。

ロールポリシーにより AWS CloudHSM 、 はユーザーに代わって Amazon CloudWatch Logs ロググループとログストリームを作成し、ログイベントを書き込むことができます。これは以下と IAM コンソールで確認できます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
```

```
    ]
}
```

AWSServiceRoleForCloudHSM ロールの信頼ポリシーにより AWS CloudHSM 、 はロールを引き受けることができます。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

サービスにリンクされたロールを作成する (自動)

AWS CloudHSM は、 AWS CloudHSM 管理者グループの作成時に定義したアクセス許可に `iam:CreateServiceLinkedRole`アクションを含めると、クラスターの作成時に AWSServiceRoleForCloudHSM ロールを作成します。「のカスタマー管理ポリシー AWS CloudHSM」を参照してください。

クラスターがすでに 1 つ以上あり、AWSServiceRoleForCloudHSM ロールを作成する場合は、コンソール、create-cluster コマンド、または API の CreateCluster オペレーションを使用してクラスターを作成します。続いて、コンソール、delete-cluster コマンド、または API の DeleteCluster オペレーションを使用してクラスターを削除します。新しいクラスターを作成すると、サービスにリンクされたロールが作成され、アカウントのすべてのクラスターに適用されます。または、ロールを手動で作成することもできます。詳細については、次の「」セクションを参照してください。

> **Note**
>
> AWSServiceRoleForCloudHSM ロールを追加するためだけにクラスターを作成する場合は、「の開始方法 AWS CloudHSM」に記載されているすべての手順を実行してクラスターを作成する必要はありません。

サービスにリンクされたロールを作成する (手動)

IAM コンソール AWS CLI、または API を使用して、AWSServiceRoleForCloudHSM ロールを作成できます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの作成」を参照してください。

サービスにリンクされたロールを編集する

AWS CloudHSM では、AWSServiceRoleForCloudHSM ロールを編集することはできません。たとえば、ロールの作成後、さまざまなエンティティが名前でロールを参照する可能性があるため、名前を変更することはできません。また、ロールのポリシーを変更することもできません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、IAM ユーザーガイドの「サービスにリンクされたロールの編集」を参照してください。

サービスリンクロールの削除

サービスにリンクされたロールは、適用されたクラスターが存在する限り削除できません。ロールを削除するには、まずクラスター内の各 HSM を削除してからクラスターを削除する必要があります。アカウント内のすべてのクラスターを削除する必要があります。その後、IAM コンソール AWS CLI、または API を使用してロールを削除できます。クラスターの削除の詳細については、「AWS CloudHSM クラスターの削除」を参照してください。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの削除」を参照してください。

のコンプライアンス検証 AWS CloudHSM

FIPS モードのクラスターの場合、 は PCI-PIN、PCI-3DS、および SOC2 のコンプライアンス要件を満たす FIPS 承認の HSMs AWS CloudHSM を提供します。 は、FIPS モード以外のクラスターを選択するオプション AWS CloudHSM もお客様に提供します。それぞれに適用される認証要件とコンプライアンス要件の詳細は、「AWS CloudHSM クラスターモード」を参照してください。

FIPS 検証済みの HSM を利用すると、 AWS クラウドのデータセキュリティに関する企業、契約、規制のコンプライアンス要件を満たすことができます。

FIPS 140-2 への準拠

連邦情報処理規格（Federal Information Processing Standards/FIPS）出版物140-2 は、機密情報を保護する暗号モジュールのセキュリティ要件を規定する米国政府のセキュリティ基準です。AWS CloudHSM hsm1.medium インスタンスタイプは FIPS 140-2 レベル 3 認定 (証明書 #4218) です。2026 年 1 月 4 日に、hsm1.medium の証明書は履歴リストに移動します。FIPS 140-3 認証 (証明書番号 4703) を取得している hsm2m.medium への移行をお勧めします。詳細については、「FIPS validation for hardware」を参照してください。

FIPS 140-3 への準拠

連邦情報処理規格（Federal Information Processing Standards/FIPS）出版物140-3 は、機密情報を保護する暗号モジュールのセキュリティ要件を規定する米国政府のセキュリティ基準です。が提供する hsm2m.medium HSMsタイプ AWS CloudHSM は、FIPS 140-3 レベル 3 認定 (証明書 #4703) です。詳細については、「FIPS validation for hardware」を参照してください。

PCI DSS コンプライアンス

ペイメントカード業界データセキュリティ基準 (PCI DSS) は、PCI Security Standards Council が管理する専有情報のセキュリティ標準です。が提供する HSMs は PCI DSS AWS CloudHSM に準拠しています。

PCI PIN コンプライアンス

PCI PIN は、ATM や POS 端末での取引に使用される個人識別番号 (PIN) データなどの情報を送信、処理、管理するためのセキュリティ要件および評価基準を提供します。が提供する hsm1.medium および hsm2m.medium HSMs AWS CloudHSM はどちらも PCI PIN に準拠しています。詳細については、「AWS CloudHSM is now PCI PIN certified」という記事を参照してください。

PCI-3DS コンプライアンス

PCI 3DS (またはスリードメインセキュア、3-D セキュア) は EMV 3D セキュア e コマース決済のデータセキュリティを提供します。PCI 3DS は、オンライン ショッピングに別のセキュリティ層を提供します。が提供する hsm1.medium および hsm2m.medium HSMs AWS CloudHSM はどちらも PCI-3DS に準拠しています。

SOC2

SOC2 は、サービス組織がクラウドとデータセンターのセキュリティ管理を実証するのに役立つフレームワークです。AWS CloudHSM は、信頼できるサービス原則に準拠するために、重要な領域に SOC2 コントロールを実装しています。詳細については、AWS SOC のよくある質問ページを参照してください。

AWS CloudHSM PCI-PIN コンプライアンスFAQs

PCI PIN は、ATM や POS 端末での取引に使用される個人識別番号 (PIN) データなどの情報を送信、処理、管理するためのセキュリティ要件および評価基準を提供します。

PCI-PIN Attestation of Compliance (AOC) および Responsibility Summary は、コンプライアンスレポートへのオンデマンドアクセス用のセルフサービスポータルである AWS Artifact を通じて入手できます。詳細については、AWS マネジメントコンソールで AWS Artifact にサインインするか、「AWS Artifact の開始方法」で詳細をご覧ください。

よくある質問

Q: コンプライアンスと責任の証明の概要 (Attestation of Compliance and Responsibility Summary) とはどのようなものですか?

準拠証明書 (AOC) は、認定 PIN 評価機関 (QPA) によって作成され、PCI-PIN 標準の該当するコントロール AWS CloudHSM を満たしています。責任概要マトリックスは、 AWS CloudHSM とその顧客のそれぞれの責任であるコントロールを記述します。

Q: コンプライアンス AWS CloudHSM 証明書を取得するにはどうすればよいですか?

PCI-PIN Attestation of Compliance (AOC) は、コンプライアンスレポートへのオンデマンドアクセス用のセルフサービスポータルである AWS Artifact を通じて入手できます。詳細については、AWS マネジメントコンソールで AWS Artifact にサインインするか、「AWS Artifact の開始方法」で詳細をご覧ください。

Q: 自分が担当している PCI PIN コントロールはどこで確認できますか?

詳細については、AWS AWS CloudHSM PCI PIN Compliance Package の「PCI PIN Responsibility Summary」を参照してください。AWS のコンプライアンスレポートにオンデマンドでアクセスするためのセルフサービスポータルである AWS Artifact を通じて、お客様が利用できます。詳細については、AWS マネジメントコンソールで AWS Artifact にサインインするか、「AWS Artifact の開始方法」で詳細をご覧ください。

Q: AWS CloudHSM 顧客として、PCI-PIN 準拠証明書 (AOC) を使用できますか?

PCI-PIN コンプライアンスはお客様自身で管理する必要があります。支払いワークロードがすべての PCI-PIN 管理/要件を満たしていることを確認するには、認定 PIN 評価者 (QPA) による正式な PCI-PIN 認証プロセスを経る必要があります。ただし、AWS が責任を負うコントロールについては、QPA AWS CloudHSM はさらにテストすることなく、準拠証明書 (AOC) に頼ることができます。

Q: キー管理ライフサイクルに関連する PCI-PIN 要件は AWS CloudHSM 担当していますか?

AWS CloudHSM は、HSMs。PCI-PIN 規格の主要な管理ライフサイクル要件はお客様の責任となります。

Q: PCI-PIN に準拠している AWS CloudHSM コントロールはどれですか?

AOC は、QPA によって評価される AWS CloudHSM コントロールを要約します。PCI-PIN Responsibility Summary は、コンプライアンスレポートへのオンデマンドアクセス用のセルフサービスポータルである AWS Artifact を通じて入手できます。

Q: PIN 翻訳や DUKPT などの支払い機能は AWS CloudHSM サポートされていますか?

いいえ。汎用 HSMs AWS CloudHSM を提供します。今後、支払い機能が提供される可能性はあります。サービスは支払い機能を直接実行しませんが、 AWS CloudHSM PCI PIN 準拠の証明により、お客様は で実行されているサービスに対して独自の PCI コンプライアンスを達成できます AWS CloudHSM。ワークロードでの AWS Payment Cryptography サービス利用に興味がある場合は、ブログ「AWS Payment Cryptography による支払い処理のクラウドへの移行」を参照してください。

非推奨通知

FIPS 140、PCI-DSS、PCI-PIN、PCI-3DS、SOC2、またはend-of-supportハードウェアの要件に準拠し続けるために、 は時折機能を廃止 AWS CloudHSM することがあります。このページには、現在適用されている変更が一覧表示されています。

HSM1 の廃止

AWS CloudHSM hsm1.medium インスタンスタイプは、2026 年 3 月 31 日にサポートが終了します。サービスを継続して利用できるよう、以下の変更を導入します。

- 2025 年 4 月以降、新しい hsm1.medium クラスターを作成できなくなります。
- 2026 年 1 月から、既存の hsm1.medium クラスターを新しい hsm2m.medium インスタンスタイプに自動的に移行します。

hsm2m.medium インスタンスタイプは、現在の AWS CloudHSM インスタンスタイプと互換性があり、パフォーマンスが向上します。アプリケーションの中断を回避するためには、クライアント SDK の最新バージョンにアップグレードする必要があります。更新する方法については、??? を参照してください。

移行には、次の 2 つの方法があります。

1. 準備ができたら、CloudHSM が管理する移行を選択します。詳細については、??? を参照してください。
2. hsm1 クラスターのバックアップから新しい hsm2m.medium クラスターを作成し、アプリケーションを新しいクラスターに切り替えます。この方法では、ブルー/グリーンデプロイ戦略の使用をお勧めします。詳細については、「???」を参照してください。

FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨

米国国立標準技術研究所 (NIST) [1] は、トリプル DES (DESede、3DES、DES3) 暗号化と PKCS #1 v1.5 パディングによる RSA キーのラップとアンラップのサポートは 2023 年 12 月 31 日以降は許可されないよう勧告しています。そのため、連邦情報処理標準 (FIPS) モードクラスターにおけるこれらのサポートは 2024 年 1 月 1 日に終了します。これらのサポートは、非 FIP モードのクラスターでも引き続きサポートされます。

このガイダンスは、以下の暗号化オペレーションに適用されます。

- トリプル DES キー生成
  - PKCS #11 ライブラリ向け `CKM_DES3_KEY_GEN`
  - JCE プロバイダー向け `DESede Keygen`
  - `genSymKey`と KMU 向け `-t=21`
- トリプル DES キーによる暗号化 (注: 復号化操作は許可されています)
  - PKCS #11 ライブラリの場合: `CKM_DES3_CBC` 暗号化、`CKM_DES3_CBC_PAD` 暗号化、`CKM_DES3_ECB` 暗号化
  - JCE プロバイダーの場合: `DESede/CBC/PKCS5Padding` 暗号化、`DESede/CBC/NoPadding` 暗号化、`DESede/ECB/Padding` 暗号化、`DESede/ECB/NoPadding` 暗号化
- PKCS #1 v1.5 パディングによる RSA キーのラップ、アンラップ、暗号化、および復号化
  - PKCS #11 SDK 向け `CKM_RSA_PKCS` ラップ、アンラップ、暗号化、および復号化
  - JCE SDK 向け `RSA/ECB/PKCS1Padding` ラップ、アンラップ、暗号化、および復号化
  - KMU 向け `-m 12` 付きの `wrapKey`と`unWrapKey` (注記12はメカニズム `RSA_PKCS` の値)

[1] この変更の詳細については、「暗号アルゴリズムとキー長の利用の変遷」の表 1 と表 5 を参照してください。

の耐障害性 AWS CloudHSM

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。復元性をサポートする AWS CloudHSM 機能の詳細については、「AWS CloudHSM クラスターの高可用性とロードバランシング」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 CloudHSM

マネージドサービスである AWS CloudHSM は、ホワイトペーパー「Amazon Web Services: セキュリティプロセスの概要」に記載されている 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。

AWS が発行した API コールを使用して、ネットワーク AWS CloudHSM 経由で にアクセスします。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。

ネットワークの隔離

仮想プライベートクラウド (VPC) は、AWS クラウド内の論理的に隔離された領域にある仮想ネットワークです。VPC のプライベートサブネットにクラスターを作成できます。VPC を作成するときにプライベートサブネットを作成できます。詳細については、「の仮想プライベートクラウド (VPC) を作成する AWS CloudHSM」を参照してください。

HSM を作成するときは、HSM とやり取りできるように、サブネットに Elastic Network Interface (ENI) HSMs AWS CloudHSM を入力します。詳細については、「AWS CloudHSM クラスターアーキテクチャ」を参照してください。

AWS CloudHSM は、クラスター内の HSMs 間のインバウンド通信とアウトバウンド通信を許可するセキュリティグループを作成します。このセキュリティグループを使用して、EC2 インスタンスが

クラスター内の HSM と通信するようにできます。詳細については、「のクライアント Amazon EC2 インスタンスセキュリティグループを設定する AWS CloudHSM」を参照してください。

ユーザーの承認

では AWS CloudHSM、HSM で実行されるオペレーションには、認証された HSM ユーザーの認証情報が必要です。詳細については、「the section called “ユーザータイプ”」を参照してください。

AWS CloudHSM および VPC エンドポイント

VPC と の間にプライベート接続を確立するには、インターフェイス VPC エンドポイント AWS CloudHSM を作成します。インターフェイスエンドポイントは、インターネットゲートウェイAWS PrivateLink、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで AWS CloudHSM APIs にプライベートにアクセスできるテクノロジーである を利用しています。VPC 内のインスタンスは、 AWS CloudHSM APIs と通信するためにパブリック IP アドレスを必要としません。VPC と AWS CloudHSM 間のトラフィックは、Amazon ネットワークを離れません。

各インターフェイスエンドポイントは、サブネット内にある 1 つ、または複数の Elastic Network Interface によって表されます。

詳細については、「Amazon VPC ユーザーガイド」の「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。

AWS CloudHSM 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 CloudHSM、Amazon VPC ユーザー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を確認してください。

- AWS CloudHSM は、VPC からのすべての API アクションの呼び出しをサポートしています。

のインターフェイス VPC エンドポイントの作成 AWS CloudHSM

Amazon VPC コンソールまたは AWS Command Line Interface () を使用して、 AWS CloudHSM サービスの VPC エンドポイントを作成できますAWS CLI。詳細については、「Amazon VPC ユーザーガイド」のインターフェイスエンドポイントの作成を参照してください。

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

```
com.amazonaws.<region>.cloudhsmv2
```

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.cloudhsmv2
```

VPC エンドポイントを使いやすくするために、VPC エンドポイントの プライベート DNS ホスト名 を有効にできます。プライベート DNS 名を有効にする オプションを選択すると、標準の AWS CloudHSM DNS ホスト名 (https://cloudhsmv2.*<region>*.amazonaws.com および https://cloudhsmv2.*<region>*.api.aws) が 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 CloudHSM ホスト名 AWS CLI を使用するため、アプリケーションやコマンドで 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、「Amazon VPC ユーザーガイド」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のアクセス」を参照してください。

の VPC エンドポイントポリシーの作成 AWS CloudHSM

VPC エンドポイントには、 AWS CloudHSMへのアクセスを制御するエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- アクションを実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントでサービスへのアクセスを制御する」を参照してください。

例: AWS CloudHSM アクションの VPC エンドポイントポリシー

以下は、 のエンドポイントポリシーの例です AWS CloudHSM。エンドポイントにアタッチすると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して、リストされた AWS CloudHSM アクションへのアクセスを許可します。その他の AWS CloudHSM アクションとそれに対応する IAM アクセス許可の ID とアクセスの管理 AWS CloudHSMについては、「」を参照してください。

```
{
   "Statement":[
      {
```

```
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "<cloudhsm>:<DescribeBackups>",
            "<cloudhsm>:<DescribeClusters>",
            "<cloudhsm>:<ListTags>",
         ],
         "Resource":"*"
      }
   ]
}
```

で管理を更新する AWS CloudHSM

AWS がファームウェアを管理します。ファームウェアはサードパーティーによってメンテナンスされます。また、hsm タイプに応じて FIPS 140-2 レベル 3 または FIPS 140-3 レベル 3 への準拠について NIST によって評価を受ける必要があります。インストールできるのは、FIPS キーによって暗号化された署名済みのファームウェアのみです (AWS にはこのキーへのアクセス権がありません)。

トラブルシューティング AWS CloudHSM

で問題が発生した場合は AWS CloudHSM、以下のトピックを参考にしてください。

トピック

- AWS CloudHSM 既知の問題
- AWS CloudHSM クライアント SDK 3 キー同期の失敗
- AWS CloudHSM クライアント SDK 3 が pkpspeed ツールで HSM のパフォーマンスを検証する
- AWS CloudHSM クライアント SDK 5 ユーザーまたはポリシーに整合性のない値が含まれている
- AWS CloudHSM クライアント SDK 5 ユーザーレプリケートの失敗
- AWS CloudHSM クライアント SDK 5 キーレプリケートの失敗
- AWS CloudHSM キーの可用性チェック中にエラーが表示される
- AWS CloudHSM JCE を使用したキーの抽出
- HSM スロットリング
- AWS CloudHSM クラスターへの接続が失われました
- CloudWatch に AWS CloudHSM 監査ログがない
- AES キーラップインの長さが非準拠IVs AWS CloudHSM
- AWS CloudHSM クラスター作成の失敗の解決
- AWS CloudHSM クライアント設定ログの取得

AWS CloudHSM 既知の問題

AWS CloudHSM には以下の既知の問題があります。詳細情報についてはトピックを選択してください。

トピック

- すべての HSM インスタンスの既知の問題
- hsm1.medium AWS CloudHSM インスタンスの既知の問題
- hsm2m.medium AWS CloudHSM インスタンスの既知の問題
- の PKCS #11 ライブラリの既知の問題 AWS CloudHSM
- JCE SDK for の既知の問題 AWS CloudHSM
- の OpenSSL Dynamic Engine の既知の問題 AWS CloudHSM

- のキーストレージプロバイダー (KSP) の既知の問題 AWS CloudHSM
- の OpenSSL プロバイダーの既知の問題 AWS CloudHSM
- の CloudHSM CLI の既知の問題 AWS CloudHSM
- で Amazon Linux 2 を実行する Amazon EC2 インスタンスの既知の問題 AWS CloudHSM
- サードパーティーアプリケーションを と統合する際の既知の問題 AWS CloudHSM
- AWS CloudHSM クラスターの変更に関する既知の問題
- hsm2.medium での AWS CloudHSM クライアントバージョン 5.12.0 を使用したオペレーション失敗の既知の問題

すべての HSM インスタンスの既知の問題

以下の問題は、key_mgmt_util コマンドラインツール、PKCS #11 SDK、JCE SDK、OpenSSL SDK のいずれを使用しているかにかかわらず、すべての AWS CloudHSM ユーザーに影響します。

トピック

- [問題]: AESキーラッピングでは、ゼロパディング付きのキーラップのスタンダード準拠の実装を提供する代わりに、PKCS#5 パディングを使用します
- 問題: クライアントデーモンがクラスターに正常に接続には、その設定ファイル少なくとも 1 つの有効な IP アドレスが必要です
- 問題: クライアント SDK 3 AWS CloudHSM を使用してハッシュおよび署名できるデータには 16 KB の上限がありました
- 問題: インポートされたキーをエクスポート不可として指定できませんでした
- 問題: key_mgmt_util の wrapKey コマンドと unWrapKey コマンドのデフォルトのメカニズムが削除されました
- 問題: クラスターに HSM が 1 つしかない場合、HSM フェイルオーバーが正しく動作しません
- 問題: クラスター内の HSM のキー容量を短期間で超えた場合、クライアントが処理されない エラー状態に陥ります
- 問題: 800 バイトを超える HMAC キーを使ったダイジェストオペレーションはサポートされていません
- 問題 : クライアント SDK 3 で配布された client_info ツールが、オプションの出力引数で指定されたパスの内容を削除します
- 問題: コンテナ化された環境で --cluster-id 引数を使用して SDK 5 設定ツールを実行すると、エラーが表示されます

- 問題: 「指定された pfx ファイルから証明書/キーを作成できませんでした。エラー: NotPkcs8」という内容のエラーが表示されます。
- 問題: SDK 5.16 以降で ECDSA 署名が「無効なメカニズム」エラーで失敗する
- 問題: 事前にハッシュ化されたデータで署名オペレーションを行うと、インタラクティブモードでセッショントークンが正しくクリアされない
- 問題: CloudHSM クライアントライブラリのデフォルトのクライアント証明書が 2026 年 1 月 31 日に期限切れになる
- 問題: CA 証明書キーが弱いため、クライアント SDK 5 接続が失敗する
- 問題: 同時実行数が多いと、暗号化オペレーションが「シーケンス番号がウィンドウ外」エラーで失敗する
- 問題: ML-DSA は FIPS モードクラスターではサポートされていません

[問題]: AESキーラッピングでは、ゼロパディング付きのキーラップのスタンダード準拠の実装を提供する代わりに、PKCS#5 パディングを使用します

さらに、パディングなしおよびゼロパディングありのキーラップはサポートされていません。

- 影響: このアルゴリズムを使用してラップおよびラップ解除しても影響はありません AWS CloudHSM。ただし、 でラップされたキーは、ページングなし仕様への準拠を期待する他の HSMs またはソフトウェア内でラップ解除 AWS CloudHSM することはできません。これは、標準に準拠したラップ解除中に、8 バイトのパディングデータがキーデータの最後に追加される可能性があるためです。外部ラップされたキーを AWS CloudHSM インスタンスに適切にラップ解除することはできません。
- 回避策: AWS CloudHSM インスタンスで PKCS #5 パディングありの AES キーラップを使用してラップされたキーを外部でラップ解除するには、キーを使用する前に余分なパディングを省きます。これを行うには、ファイルエディターで余分なバイトをトリミングするか、コード内の新しいバッファにキーバイトだけをコピーします。
- 解決策のステータス: 3.1.0 クライアントおよびソフトウェアリリースで、 AWS CloudHSM に AES キーラップの標準に準拠したオプションが用意されています。詳細については、「AES キーラップ」を参照してください。

問題: クライアントデーモンがクラスターに正常に接続には、その設定ファイル少なくとも 1 つの有効な IP アドレスが必要です

- 影響: クラスター内のすべての HSM を削除し、新しい IP アドレスを取得する別の HSM を追加した場合、クライアントデーモンは元の IP アドレスで HSM を検索し続けます。
- 解決策: 断続的なワークロードを実行する場合、CreateHsm の関数で `IpAddress` 引数を使用して Elastic Network Interface (ENI) を元の値に設定することをお勧めします。ENI はアベイラビリティーゾーン (AZ) に固有である点に注意してください。代わりに、`/opt/cloudhsm/daemon/1/cluster.info` ファイルを削除した後、新しい HSM クライアントの IP アドレスにクライアント設定をリセットできます。`client -a <IP address>` コマンドを使用できます。詳細については、「クライアントのインストールと設定 (Linux)」または AWS CloudHSM「AWS CloudHSM クライアントのインストールと設定 (Windows)」を参照してください。

問題: クライアント SDK 3 AWS CloudHSM を使用してハッシュおよび署名できるデータには 16 KB の上限がありました

- 解決策のステータス: サイズが 16 KB 未満のデータは、ハッシュ用に引き続き HSM に送信されます。16 ~ 64 KB のサイズのデータをローカルやソフトウェアでハッシュする機能が追加されました。データバッファが 64 KB を超える場合、クライアント SDK 5 は明示的に失敗します。この修正を適用するには、クライアントと SDK を 5.0.0 より新しいバージョンに更新する必要があります。

問題: インポートされたキーをエクスポート不可として指定できませんでした

- 解決策のステータス: この問題は修正されています。修正を反映させるためにお客様側で必要なアクションはありません。

問題: key_mgmt_util の wrapKey コマンドと unWrapKey コマンドのデフォルトのメカニズムが削除されました

- 解決策: wrapKey コマンドまたは unWrapKey コマンドを使用する場合は、`-m` オプションを使用してメカニズムを指定する必要があります。詳細については、wrapKey または unWrapKey の記事の例を参照してください。

問題: クラスターに HSM が 1 つしかない場合、HSM フェイルオーバーが正しく動作しません

- 影響: クラスター内に 1 つしかない HSM インスタンスの接続が失われると、後で回復しても、クライアントはインスタンスに再接続しません。
- 回避方法: 本番稼働用クラスターに少なくとも 2 つの HSM インスタンスを用意することをお勧めします。この構成を使用すれば、この問題の影響を受けません。HSM が 1 つしかないクラスターの場合、クライアントデーモンをバウンスして接続を復元します。
- 解決策のステータス: この問題は、 AWS CloudHSM クライアント 1.1.2 のリリースで解決されています。修正のメリットを享受するには、このクライアントにアップグレードする必要があります。

問題: クラスター内の HSM のキー容量を短期間で超えた場合、クライアントが処理されない エラー状態に陥ります

- 影響: クライアントが処理されないエラー状態になると、フリーズし、再起動が必要になります。
- 回避方法: スループットをテストして、クライアントが処理できない速さでセッションキーを作成していないか、確認します。速さを落とすには、クラスターに HSM を追加するか、セッションキーの作成を遅くします。
- 解決策のステータス: この問題は、 AWS CloudHSM クライアント 1.1.2 のリリースで解決されています。修正のメリットを享受するには、このクライアントにアップグレードする必要があります。

問題: 800 バイトを超える HMAC キーを使ったダイジェストオペレーションはサポートされていません

- 影響: 800 バイトを上回る HMAC キーが HSM で生成されたり、HSM にインポートされたりする可能性があります。ただし、このような大きなキーを JCE または key_mgmt_util を介してダイジェストオペレーションに使用すると、オペレーションが失敗します。PKCS11 を使用している場合、HMAC キーのサイズは 64 バイトに制限されます。
- 回避方法: HSM のダイジェストオペレーションに HMAC キーを使用する場合は、必ずサイズが 800 バイト以下のものを使用します。
- 解決策のステータス: 現時点ではありません。

問題：クライアント SDK 3 で配布された client_info ツールが、オプションの出力引数で指定されたパスの内容を削除します

- 影響: 指定した出力パスの下にある既存のファイルとサブディレクトリはすべて、永久に失われる可能性があります
- 防止策: `-output` *`path`* ツールを使用する際、オプションの引数 `client_info` を使用しないでください。
- 解決策の現状: この問題は、クライアント SDK 3.3.2 のリリース によって解決されています。修正のメリットを享受するには、このクライアントにアップグレードする必要があります。

問題: コンテナ化された環境で **`--cluster-id`** 引数を使用して SDK 5 設定ツールを実行すると、エラーが表示されます

設定ツールで--cluster-id 引数を使用すると、次のエラーが表示されます。

```
No credentials in the property bag
```

このエラーは、インスタンスメタデータサービスのバージョン 2 (IMDSv2) の更新が原因で発生します。詳細については、「IMDSv2」のドキュメントを参照してください。

- 影響: この問題は、コンテナ化された環境で SDK バージョン 5.5.0 以降の設定ツールを実行し、EC2 インスタンスメタデータを利用して認証情報を提供するユーザーに影響を及ぼします。
- 回避策: PUT レスポンスのホップ制限を 2 以上に設定します。この設定方法については、Configure the instance metadata options を参照してください。

問題: 「指定された pfx ファイルから証明書/キーを作成できませんでした。エラー: NotPkcs8」という内容のエラーが表示されます。

- 回避策: openssl コマンドを使用して、カスタム SSL プライベートキーを PKCS8 形式に変換できます。 `openssl pkcs8 -topk8 -inform PEM -outform PEM -in` *`ssl_private_key`* `-out` *`ssl_private_key_pkcs8`*
- 解決策の現状: この問題は、クライアント SDK 5.12.0 のリリース によって解決されています。この修正のメリットを享受するには、このクライアント バージョン以降にアップグレードする必要があります。

問題: SDK 5.16 以降で ECDSA 署名が「無効なメカニズム」エラーで失敗する

- 影響: キー強度より弱いハッシュ関数を使用した場合、ECDSA 署名オペレーションが失敗します。この失敗は、FIPS 186-5 により「ハッシュ関数はキー強度と同等以上である必要がある」と定められているために発生します。

  クライアントログには、次のようなエラーが表示される場合があります。

  ```
  [cloudhsm_provider::hsm1::session::ecdsa::sign::common][][] Digest security strength
   (80) is weaker than the key security strength (128)
  ```

- 回避策: ハッシュ関数を更新できない場合は、ハッシュ強度要件を適用しない 非 FIPS クラスターへ移行することができます。ただし、FIPS 準拠を維持するためには、ハッシュ関数を更新することをお勧めします。

  さらに追加の回避策として、この要件をバイパスするための設定オプションを追加しています。ただし、弱いハッシュ関数を使用した ECDSA 署名はセキュリティのベストプラクティスに反するため、このオプションは推奨されません。このオプションを使用するには、次のコマンドを実行します (使用している SDK に応じて `configure-cli` を適切な設定ツールに置き換えてください: the section called “構文”)。

  ```
  sudo /opt/cloudhsm/bin/configure-cli --enable-ecdsa-with-weak-hash-function
  ```

- 解決策: ECDSA キーの強度と同等以上の強度を持つハッシュ関数を使用してください。ハッシュ関数と ECDSA キー強度の詳細については、NIST SP 800-57 Part 1 Rev 5 の表 2 および表 3 を参照してください。

問題: 事前にハッシュ化されたデータで署名オペレーションを行うと、インタラクティブモードでセッショントークンが正しくクリアされない

- 影響: CloudHSM CLI をインタラクティブモードで使用している場合、SDK バージョン 5.16.1 において、事前にハッシュ化されたデータを使用した署名オペレーションが、セッショントークンを正しくクリアできません。
- 回避策: 事前にハッシュ化されたデータで署名オペレーションを行う際は、インタラクティブモードではなく単一コマンドモードを使用してください。これにより、各オペレーションの後にトークンが正しくクリーンアップされます。
- 解決策のステータス: この問題は、CloudHSM SDK 5.16.2 で解決されています。修正を適用するには、バージョン 5.16.2 以降にアップグレードしてください。

問題: CloudHSM クライアントライブラリのデフォルトのクライアント証明書が 2026 年 1 月 31 日に期限切れになる

- 影響: 2026 年 1 月 31 日以降、お客様への影響はありません。クライアントと HSM 間のすべての通信は、ここで説明するように、クライアント HSM TLS によって保護されます。クライアント HSM TLS は、クラスターの初期化中に顧客が設定した顧客所有/管理の証明書を使用します。
- 解決ステータス: Resolved。

問題: CA 証明書キーが弱いため、クライアント SDK 5 接続が失敗する

Client SDK 5 バージョン 5.17.0 または 5.17.1 にアップグレードした後、アプリケーションは次のエラーで AWS CloudHSM クラスターに接続できません。

```
Certificate verification error with error code 67 and depth 1: CA certificate key too
 weak
```

- 影響: 2048 ビットより弱い RSA キーを使用して CA 証明書で初期化されたクラスターは、クライアント SDK 5 バージョン 5.17.0 および 5.17.1 で接続に失敗します。
- 回避策: 5.17.0 より前のバージョンのクライアント SDK 5 にダウングレードして、クラスターへの接続を復元します。
- 解決ステータス: この問題はクライアント SDK 5.17.2 で解決されています。修正を活用するには、バージョン 5.17.2 以降にアップグレードします。

問題: 同時実行数が多いと、暗号化オペレーションが「シーケンス番号がウィンドウ外」エラーで失敗する

- 影響: 同時実行性の高いワークロードでは、クライアント SDK 5 が一時的なオペレーションの失敗を返すことがあります。クライアント SDK 5 は、セキュリティ体制の向上の一環として、HSM への並列オペレーションの量を増やします。負荷が高い場合、スレッドスケジューリングの遅延により、再試行時に回復可能な一時的な障害が発生する可能性があります。
- 回避策: 一時的なエラーに対してアプリケーションレベルの再試行ロジックを実装します。ベストプラクティスガイドを参照してください。また、並列暗号化オペレーションの数を制限するか、より多くの CPU コアを持つインスタンスタイプにアップグレードすることで、クライアントの負荷を減らすこともできます。

- 解決ステータス: クライアント SDK 5.17.2 に再試行が追加されました。更新を利用するには、このクライアントバージョン 以降にアップグレードする必要があります。

問題: ML-DSA は FIPS モードクラスターではサポートされていません

- 影響: ML-DSA (Module-Lattice-Based Digital Signature Algorithm) のキー生成、署名、検証は現在、FIPS モードクラスターではサポートされていません。
- 解決策: この問題は解決されました。お客様は、 AWS CloudHSM クライアント SDK 5.18 の FIPS モードと非 FIPS モードの両方で ML-DSA を使用できるようになりました。

hsm1.medium AWS CloudHSM インスタンスの既知の問題

以下の問題は、すべての AWS CloudHSM hsm1.medium インスタンスに影響します。

トピック

- 問題: HSM が 250 人を超えるユーザーを作成できない

問題: HSM が 250 人を超えるユーザーを作成できない

- 回避策: この問題は hsm2m.medium AWS CloudHSM インスタンスタイプで解決されます。
- 解決策のステータス: 現時点ではありません。

hsm2m.medium AWS CloudHSM インスタンスの既知の問題

以下の問題は、すべての AWS CloudHSM hsm2m.medium インスタンスに影響します。

トピック

- 問題: hsm2m.medium でログイン遅延が増加する
- 問題: hsm2m.medium で find key の遅延が増加する
- 問題: キーの信頼された属性を設定しようとする CO は、クライアント SDK 5.12.0 以前では失敗します。
- 問題: ECDSA 検証は、FIPS モードでクラスターのクライアント SDK 5.12.0 以前では失敗します
- 問題: PEM 形式の証明書のみが CloudHSM CLI で mtls トラストアンカーとして登録できます
- 問題: パスフレーズで保護されたクライアントプライベートキーを使用して mTLS を利用すると、アプリケーションがすべてのリクエスト処理を停止します。

- 問題: CloudHSM CLI を使用するとユーザーのレプリケートが失敗する
- 問題: バックアップの作成中にオペレーションが失敗することがある
- 問題: hsm2m.medium の一部のシナリオで、クライアント SDK 5.8 以降が HSM のスロットルされたオペレーションに対して自動リトライを実行しない
- 問題: hsm2m.medium ですべての IV がゼロの AES/CBC ラップ解除オペレーションが失敗する
- 問題: hsm2m.medium でのアプリケーションのコールドスタート中に HSM 接続を初期化できない

問題: hsm2m.medium でログイン遅延が増加する

- 影響: hsm2m.medium にログインすると、コンプライアンス要件が過度に厳密に解釈されるため、レイテンシーが増加します。
- 解決策: 2025 年 12 月 20 日より前に新しい hsm2m.medium インスタンスを作成した場合、または hsm1.medium から hsm2m.medium に移行した場合は、パスワードをリセットして、ログインオペレーション用に実装されたパフォーマンスの向上を活用する必要があります。手順については、変更パスワードを参照してください。

問題: hsm2m.medium で find key の遅延が増加する

- 影響: hsm2m.medium の HSM インスタンスは、hsm1.medium と比較して、より一貫性のある予測可能なパフォーマンスを提供する改良されたフェアシェアアーキテクチャを備えています。一方、hsm1.medium では、HSM リソースの不規則な使用により find key の処理が高速に見える場合があります。しかし、hsm1.medium の find key パフォーマンスは、HSM インスタンスがパッチ適用または新しいファームウェアに更新されると低下します。この問題は、JCE の `KeyStore.getKey()` などのオペレーションに影響します。
- 解決策: この問題は解決されました。ベストプラクティスとして、find key オペレーションの結果をキャッシュしてください。find key は HSM 内でリソース集約的なオペレーションであるため、キャッシュにより実行回数を減らすことができます。さらに、エクスポネンシャルバックオフとジッターを用いたクライアント側のリトライを実装し、HSM のスロットリング失敗を減らしてください。

問題: キーの信頼された属性を設定しようとする CO は、クライアント SDK 5.12.0 以前では失敗します。

- 影響: キーの信頼された属性を設定しようとする CO ユーザーは、そのことを示すエラーを受け取りますUser type should be CO or CU。
- 解決策: クライアント SDK の将来のバージョンでは、この問題が解決されます。更新については、ユーザーガイドの「ドキュメント履歴」で発表されます。

問題: ECDSA 検証は、FIPS モードでクラスターのクライアント SDK 5.12.0 以前では失敗します

- 影響: FIPS モードで HSM に対して実行される ECDSA 検証オペレーションが失敗します。
- 解決策の現状: この問題は、クライアント SDK 5.13.0 のリリースによって解決されています。この修正のメリットを享受するには、このクライアント バージョン以降にアップグレードする必要があります。

問題: PEM 形式の証明書のみが CloudHSM CLI で mtls トラストアンカーとして登録できます

- 影響: DER 形式の証明書は、CloudHSM CLI で mTLS トラストアンカーとして登録できません。
- 回避策: openssl コマンドを使用して、DER 形式の証明書を PEM 形式に変換できます。 `openssl x509 -inform DER -outform PEM -in certificate.der -out certificate.pem`

問題: パスフレーズで保護されたクライアントプライベートキーを使用して mTLS を利用すると、アプリケーションがすべてのリクエスト処理を停止します。

- 影響: アプリケーションが実行するすべてのオペレーションが停止し、アプリケーションの実行中にユーザーへ標準入力でパスフレーズの入力を複数回要求されます。パスフレーズがオペレーションのタイムアウト前に入力されない場合、その操作はタイムアウトし、失敗します。
- 回避策: mTLS ではパスフレーズで暗号化されたプライベートキーはサポートされていません。クライアントプライベートキーからパスフレーズ暗号化を削除してください。

問題: CloudHSM CLI を使用するとユーザーのレプリケートが失敗する

- 影響: CloudHSM CLI を使用した場合、hsm2m.medium インスタンスではユーザーレプリケートが失敗します。`user replicate` コマンドは hsm1.medium インスタンスでは正常に動作します。
- 解決策: この問題は解決されました。

問題: バックアップの作成中にオペレーションが失敗することがある

- 影響: AWS CloudHSM がバックアップを作成している間、hsm2m.medium インスタンスでは、乱数生成などのオペレーションが失敗する可能性があります。
- 解決策: サービス中断を最小限に抑えるため、以下のベストプラクティスを実装してください。
  - 複数の HSM からなるクラスターを作成する
  - アプリケーション側でクラスターオペレーションのリトライを実装する

  ベストプラクティスの詳細については、「のベストプラクティス AWS CloudHSM」を参照してください。

問題: hsm2m.medium の一部のシナリオで、クライアント SDK 5.8 以降が HSM のスロットルされたオペレーションに対して自動リトライを実行しない

- 影響: クライアント SDK 5.8 以降では、特定の HSM スロットルオペレーションが自動的にリトライされません
- 回避策: 負荷に耐えられるようクラスターを設計し、アプリケーション側でリトライ処理を実装するというベストプラクティスに従ってください。現在、この問題の修正に取り組んでいます。更新については、ユーザーガイドの「ドキュメント履歴」で発表されます。
- 解決ステータス: この問題は AWS CloudHSM クライアント SDK 5.16.2 で解決されています。この修正のメリットを享受するには、このクライアント バージョン以降にアップグレードする必要があります。

問題: hsm2m.medium ですべての IV がゼロの AES/CBC ラップ解除オペレーションが失敗する

- 影響: AWS CloudHSM JCE プロバイダーを使用してキーをラップ解除するために AES/CBC メカニズムを使用する場合、hsm1.medium インスタンスにない検証チェックが追加されているため、16 バイトのゼロフィル IV を使用したオペレーションは hsm2m.medium インスタンスで失敗します。

- 解決ステータス: AES/CBC ラップ解除オペレーション中にゼロバイト IVs を受け入れることができる修正に取り組んでいます。

問題: hsm2m.medium でのアプリケーションのコールドスタート中に HSM 接続を初期化できない

- 影響: この問題は、クライアントアプリケーションのデプロイや再起動などのコールドスタートに影響します。hsm2m.medium HSM インスタンスは公平配分アーキテクチャを改善し、すべてのお客様に一貫したパフォーマンス、スループット、レイテンシーを提供します。現在、hsm1.medium では、HSM 接続の同時初期化で意図したよりも高いパフォーマンスが見られる場合があります。ただし、hsm1.medium 接続の初期化パフォーマンスは、基盤となるシステム更新によって異なります。
- 解決策: ベストプラクティスに従い、クライアントアプリケーションのデプロイと再起動をずらして、HSM 接続を同時に初期化するクライアントアプリケーションの量を制限します。また、クライアントアプリケーションの初期化のためにアプリケーションレベルの再試行を実装することをお勧めします。さらに、 で設定ツールを使用してブートストラップ`--cluster-id <cluster ID>`し、すべての HSM IP をクライアント設定ファイルに追加します。この動作は、AWS CloudHSM Client SDK バージョン 5.17.1 以降で改善されました。この改善の恩恵を受けるには、最新の SDK バージョンにアップグレードすることをお勧めします。

の PKCS #11 ライブラリの既知の問題 AWS CloudHSM

次の問題は、 の PKCS #11 ライブラリに影響します AWS CloudHSM。

トピック

- 問題: PKCS #11 ライブラリのバージョン 3.0.0 での AES キーラップが、使用前に IV を検証しません
- 問題: PKCS #11 SDK 2.0.4 以前のバージョンでは、AES キーのラップとラップ解除に常に 0xA6A6A6A6A6A6A6A6 のデフォルトの IV が使用されていました。
- 問題: CKA_DERIVE 属性はサポート外だったため、処理されませんでした
- 問題: CKA_SENSITIVE 属性はサポート外だったため、処理されませんでした
- 問題: マルチパートのハッシュおよび署名がサポートされていません
- 問題: C_GenerateKeyPair は、プライベートテンプレートで標準に従った方法では、CKA_MODULUS_BITS や CKA_PUBLIC_EXPONENT を処理しません

- 問題: C_Encrypt メカニズムを使用している場合、 C_Decrypt および CKM_AES_GCM API オペレーションのバッファが 16 KB を超えることができません
- 問題: 楕円曲線ディフィーヘルマン (ECDH) キーの導出が、HSM 内で部分的に実行されます
- 問題: CentOS6 や RHEL 6 などの EL6 プラットフォームで secp256k1 署名の検証が失敗します
- 問題 : 関数呼び出しの順序が正しくないと、失敗する代わりに未定義の結果が得られる。
- 問題 : SDK 5 では読み取り専用セッションは対応していません
- 問題: cryptoki.hヘッダーファイルは Windows 専用です
- 問題: スロットリングCKR_DEVICE_ERROR時に C_GetAttributeValue が返されます
- 問題: PKCS #11 ライブラリがスロットリングされた証明書ストレージオペレーションを再試行しない

問題: PKCS #11 ライブラリのバージョン 3.0.0 での AES キーラップが、使用前に IV を検証しません

長さが 8 バイトより短い IV を指定すると、使用前に予測不可能なバイトが埋め込まれます。

> **Note**
>
> これは `CKM_AES_KEY_WRAP` メカニズムがある `C_WrapKey` にのみ影響します。

- 影響: PKCS #11 バージョン 3.0.0 で 8 バイトより短い IV を指定した場合、キーをラップ解除できない可能性があります。
- 回避方法:
  - PKCS #11 バージョン 3.0.1 以降にアップグレードすることを強くお勧めします。これにより、AES キーラップ時に IV の長さが適切に適用されます。ラップコードを修正して NULL IV を渡すか、`0xA6A6A6A6A6A6A6A6` のデフォルトの IV を指定します。詳細情報は、「トラブルシューティングガイド : AESキーラップ用非対応長さのCustom IV」を参照してください。
  - 8 バイトより短い IV を使用して PKCS #11 ba-jon 3.0.0 でキーをラップした場合は、弊社 サポートデスク へご連絡ください。
- 解決策のステータス: この問題は PKCS #11 SDK バージョン 3.0.1 で解決されています。AES キーラップを使用してキーをラップするには、NULL または 8 バイトの長さの IV を指定します。

問題: PKCS #11 SDK 2.0.4 以前のバージョンでは、AES キーのラップとラップ解除に常に **0xA6A6A6A6A6A6A6A6** のデフォルトの IV が使用されていました。

ユーザーが指定した IV はそのまま無視されていました。

> **Note**
>
> これは CKM_AES_KEY_WRAP メカニズムがある C_WrapKey にのみ影響します。

- 影響:
  - PKCS #11 SDK 2.0.4 以前のバージョンとユーザーが指定した IV を使用した場合、キーは 0xA6A6A6A6A6A6A6A6 のデフォルトの IV でラップされます。
  - PKCS #11 SDK 3.0.0 以降とユーザーが指定した IV を使用した場合、キーはユーザーが指定した IV でラップされます。
- 回避方法:
  - PKCS #11 SDK 2.0.4 以前でラップされたキーをラップ解除するには、0xA6A6A6A6A6A6A6A6 のデフォルトの IV を使用します。
  - PKCS #11 SDK 3.0.0 以降でラップされたキーをラップ解除するには、ユーザーが指定した IV を使用します。
- 解決策のステータス: ラップおよびラップ解除コードを修正して NULL IV を渡すか、0xA6A6A6A6A6A6A6A6 のデフォルトの IV を指定することを強くお勧めします。

問題: **CKA_DERIVE** 属性はサポート外だったため、処理されませんでした

- 解決策のステータス: FALSE が設定されている場合は、CKA_DERIVE を受け付けるように修正を実装し ます。 AWS CloudHSMにキー取得関数サポートが追加されるまで、TRUE に設定された CKA_DERIVE はサポートされません。この修正を適用するには、クライアントと SDK をバージョン 1.1.1 以上に更新する必要があります。

問題: **CKA_SENSITIVE** 属性はサポート外だったため、処理されませんでした

- 解決策のステータス: CKA_SENSITIVE 属性を受け入れ、適切に遵守するように修正を実装しました。この修正を適用するには、クライアントと SDK をバージョン 1.1.1 以上に更新する必要があります。

問題: マルチパートのハッシュおよび署名がサポートされていません

- 影響: C_DigestUpdate および C_DigestFinal は実装されません。C_SignFinal も実装されていないため、NULL 以外のバッファでは CKR_ARGUMENTS_BAD で エラーが発生します。
- 回避策: アプリケーション内でデータをハッシュし、ハッシュの署名 AWS CloudHSM にのみ使用します。
- 解決策のステータス: クライアントと SDK を修正し、マルチパートハッシュを正しく実装する予定です。更新は AWS CloudHSM フォーラムとバージョン履歴ページで告知されます。

問題: **C_GenerateKeyPair** は、プライベートテンプレートで標準に従った方法では、**CKA_MODULUS_BITS** や **CKA_PUBLIC_EXPONENT** を処理しません

- 影響: プライベートテンプレートに または C_GenerateKeyPair が含まれている場合、CKA_TEMPLATE_INCONSISTENTCKA_MODULUS_BITS は CKA_PUBLIC_EXPONENT を返します。代わりに、すべてのフィールドが FALSE に設定されているプライベートキーを生成します。キーは使用できません。
- 解決策: アプリケーションによって、エラーコードに加えて使用状況フィールドの値をチェックすることをお勧めします。
- 解決策のステータス: 修正を実装し、間違ったプライベートキーテンプレートが使用されている場合に適切なエラーメッセージを返すようにします。更新された PKCS#11 ライブラリは、バージョン履歴ページで告知されます。

問題: **C_Encrypt** メカニズムを使用している場合、 **C_Decrypt** および **CKM_AES_GCM** API オペレーションのバッファが 16 KB を超えることができません

AWS CloudHSM はマルチパート AES-GCM 暗号化をサポートしていません。

- 影響: CKM_AES_GCM メカニズムを使用して 16 KB を超えるデータを暗号化することができません。
- 回避策: CKM_AES_CBC、CKM_AES_CBC_PAD などの代替メカニズムを使用するか、データを複数部分に分割し、AES_GCM を使用して各部分を個別に暗号化できます。を使用している場合はAES_GCM、データの分割とその後の暗号化を管理する必要があります。 AWS CloudHSM はマルチパート AES-GCM 暗号化を実行しません。FIPS では、AES-GCM の初期化ベクター (IV) を HSM で生成する必要があります。したがって、AES-GCM 暗号化データの各部分の IV は異なります。

- 解決策のステータス: SDK を修正し、データバッファが大きすぎる場合は明示的に失敗するようにします。`C_EncryptUpdate` および `C_DecryptUpdate` API オペレーションに `CKR_MECHANISM_INVALID` を返します。マルチパート暗号化を使用しなくても大きいバッファをサポートできる代替方法を評価しています。更新は、 AWS CloudHSM フォーラムとバージョン履歴ページで発表されます。

問題: 楕円曲線ディフィーヘルマン (ECDH) キーの導出が、HSM 内で部分的に実行されます

EC プライベートキーは常に HSM にありますが、キーの取得手順は複数のステップで実行されます。その結果、各ステップの中間結果がクライアントに存在します。

- 影響: クライアント SDK 3 では、`CKM_ECDH1_DERIVE` メカニズムを使用して得られたキーは、まずクライアントで使用可能になり、その後 HSM 内にインポートされます。その後、キーのハンドルがアプリケーションに返されます。
- 回避策: AWS CloudHSMに SSL/TLS のオフロードを実装すると、この制限が問題とはならない場合があります。アプリケーションでキーが常に FIPS 境界内に留まる必要がある場合、ECDH キー取得に依存しない代替プロトコルの使用を検討してください。
- 解決策のステータス: SDK 5.16 は、ECDH とキー導出をサポートしており、これらの処理はすべて HSM 内で実行されます。

問題: CentOS6 や RHEL 6 などの EL6 プラットフォームで secp256k1 署名の検証が失敗します

これは、CloudHSM PKCS#11 ライブラリが、OpenSSL を使用して EC 曲線データを検証することにより、検証操作の初期化中にネットワークの呼び出しを回避するために発生します。Secp256k1 は EL6 プラットフォームのデフォルトの OpenSSL パッケージでサポートされていないため、初期化は失敗します。

- 影響: Secp256k1 署名検証が EL6 プラットフォームで失敗します。検証呼び出しは、`CKR_HOST_MEMORY` エラーで失敗します。
- 回避策: PKCS#11 アプリケーションで secp256k1 の署名を検証する必要がある場合は、Amazon Linux 1 または任意の EL7 プラットフォームを使用することをお勧めします。または、secp256k1 曲線 をサポートする OpenSSL パッケージのバージョンにアップグレードすることもできます。
- 解決策のステータス: ローカル曲線検証が利用できない場合に HSM にフォールバックするための修正を実装中です。更新された PKCS#11 ライブラリは、バージョン履歴ページで告知されます。

問題：関数呼び出しの順序が正しくないと、失敗する代わりに未定義の結果が得られる。

- 影響：誤った一連の関数を呼び出すと、個々の関数呼び出しの返しが成功しても、最終的な結果は正しくありません。例えば、復号化されたデータが元のプレーンテキストと一致しない場合や、署名が検証できない場合があります。この問題は、シングルパートとマルチパートの両方のオペレーションに影響します。

  次の表は、PKCS #11 オペレーションの正しい関数呼び出しシーケンスを示しています。

  単一パートオペレーションの正しいシーケンス:

運用	正しいシーケンス
Encrypt	C_EncryptInit → C_Encrypt
Decrypt	C_DecryptInit → C_Decrypt
Sign	C_SignInit → C_Sign
検証	C_VerifyInit → C_Verify
オブジェクトの検索	C_FindObjectsInit → C_FindObjects → C_FindObjectsFinal

  マルチパートオペレーションの正しいシーケンス:

運用	正しいシーケンス
暗号化	C_EncryptInit → C_EncryptUpdate (1 つ以上) → C_EncryptFinal
Decrypt	C_DecryptInit → C_DecryptUpdate (1 つ以上) → C_DecryptFinal
Sign	C_SignInit → C_SignUpdate (1 つ以上) → C_SignFinal

運用	正しいシーケンス
検証	C_VerifyInit → C_VerifyUpdate (1 つ以上) → C_VerifyFinal

- 回避策: アプリケーションは、PKCS #11 仕様に従って、上記の表に示すように、単一パートオペレーションとマルチパートオペレーションの両方に正しい一連の関数呼び出しを使用する必要があります。この状況では、アプリケーションが CloudHSM PKCS #11 ライブラリに依存してエラーを返す必要はありません。

問題 : SDK 5 では読み取り専用セッションは対応していません

- 問題: SDK 5 では、C_OpenSession で読み取り専用セッションを開くことはできません。
- 影響: C_OpenSession 未対応で CKF_RW_SESSION を呼び出ししようとした場合、呼び出しは失敗し、エラー CKR_FUNCTION_FAILED が返されます。
- 防止策: セッションを開く際、CKF_SERIAL_SESSION | CKF_RW_SESSION 関数呼び出しに C_OpenSession フラグを渡す必要があります。

問題: **cryptoki.h**ヘッダーファイルは Windows 専用です

- 問題: Linux の AWS CloudHSM クライアント SDK 5 バージョン 5.0.0 から 5.4.0 では、ヘッダーファイルは Windows オペレーティングシステムとのみ互換性/opt/cloudhsm/include/pkcs11/cryptoki.hがあります。
- 影響: Linux ベースのオペレーティングシステム上のアプリケーションにこのヘッダーファイルを含めようとすると、問題が発生する可能性があります。
- 解決ステータス: このヘッダーファイルの Linux 互換バージョンを含む AWS CloudHSM クライアント SDK 5 バージョン 5.4.1 以降にアップグレードします。

問題: スロットリング**CKR_DEVICE_ERROR**時に **C_GetAttributeValue** が返されます

C_GetAttributeValue 呼び出しがスロットリングされると、 が返されますCKR_DEVICE_ERROR。他のスロットリングされたオペレーションは を返しますCKR_FUNCTION_FAILED。

- 影響: アプリケーションで でのみ再試行する場合CKR_FUNCTION_FAILED、スロットリングされたリクエストは再試行されません。
- 回避策: CKR_DEVICE_ERRORと の両方でエクスポネンシャルバックオフを使用してスロットリングされたオペレーションを再試行しますCKR_FUNCTION_FAILED。スロットリングが続く場合は、クラスターに HSMs を追加して、読み取りと書き込みの合計レートを上げるか、リクエストレートを減らします。詳細については、「HSM スロットリング」を参照してください。
- 解決ステータス: この問題はクライアント SDK 5.18.0 で解決されました。 はスロットリングされた証明書ストアとキー属性の読み取りCKR_FUNCTION_FAILEDに対して を返すC_GetAttributeValueようになりました。修正を活用するには、バージョン 5.18.0 以降にアップグレードします。

問題: PKCS #11 ライブラリがスロットリングされた証明書ストレージオペレーションを再試行しない

証明書ストレージオペレーションのレート制限を超えると、PKCS #11 ライブラリはこれらのスロットリングされたオペレーションを自動的に再試行しません。

- 影響: スロットリングされた証明書ストレージオペレーションは、自動的に再試行されるのではなく、アプリケーションにエラーを返します。
- 回避策: エクスポネンシャルバックオフを使用して、アプリケーションでスロットリングされた証明書ストレージオペレーションを再試行します。スロットリングが続く場合は、クラスターに HSMs を追加して、読み取りと書き込みの合計レートを上げるか、リクエストレートを減らします。詳細については、「HSM スロットリング」を参照してください。
- 解決ステータス: この問題はクライアント SDK 5.18.0 で解決されています。PKCS #11 ライブラリは、スロットリングされた証明書ストレージオペレーションを自動的に再試行するようになりました。修正を活用するには、バージョン 5.18.0 以降にアップグレードします。

JCE SDK for の既知の問題 AWS CloudHSM

次の問題は JCE SDK for に影響します AWS CloudHSM。

トピック

- 問題: 非対称キーペアを使用する場合、明示的にキーを作成またはインポートしない場合でも、キー容量が占有されます。
- 問題: JCE KeyStore は読み取り専用です

- 問題: AES-GCM 暗号化のバッファが 16,000 バイトを超えることはできません
- 問題: 楕円曲線ディフィーヘルマン (ECDH) キーの導出が、HSM 内で部分的に実行されます
- 問題: KeyGenerator と KeyAttribute がキーサイズパラメータをビット数ではなくバイト数として誤って解釈します
- 問題: クライアント SDK 5 から「不正な反射アクセスオペレーションが発生しました」という警告が表示されます
- 問題: JCE セッションプールが使い果たされています
- 問題: getKey 操作によるクライアント SDK 5 メモリリーク

問題: 非対称キーペアを使用する場合、明示的にキーを作成またはインポートしない場合でも、キー容量が占有されます。

- 影響: この問題により、HSM が予期せずキー領域を使い果たすことがあります。この問題は、アプリケーションが暗号化オペレーションに `CaviumKey` オブジェクトではなく標準の JCE キーオブジェクトを使用する場合に発生します。標準の JCE キーオブジェクトを使用する場合、`CaviumProvider` によって暗黙的にそのキーが HSM にインポートされ、アプリケーションが終了するまでセッションキーによってこのキーは削除されません。その結果、アプリケーションの実行中にキーが蓄積され、HSM の空きキー領域が不足して、アプリケーションがフリーズする可能性があります。
- 回避策: `CaviumSignature` クラス、`CaviumCipher` クラス、`CaviumMac` クラス、または `CaviumKeyAgreement` クラスを使用する場合は、標準の JCE キーオブジェクトではなく `CaviumKey` としてキーを指定してください。

  `CaviumKey` クラスを使用して通常のキーを `ImportKey` に手動で変更し、オペレーションの完了後にキーを手動で削除できます。
- 解決策のステータス: 暗黙的なインポートを適切に管理するために、`CaviumProvider` を更新中です。修正は、利用可能なバージョン履歴ページで告知されます。

問題: JCE KeyStore は読み取り専用です

- 影響: 現在、HSM でサポートされていないオブジェクトタイプを JCE KeyStore に保存することはできません。具体的には、キーストアに証明書を保存することはできません。これにより、キーストア内で証明書を見つけることを期待する jarsigner などのツールとの相互運用性が妨げられます。

- [Workaround(回避策)]: キーストアではなく、ローカルファイルまたは S3 バケットの場所から証明書をロードするようにコードを修正することができます。
- 解決ステータス: AWS CloudHSM キーストアを使用して証明書を保存できます。

問題: AES-GCM 暗号化のバッファが 16,000 バイトを超えることはできません

マルチパート AES-GCM 暗号化は対応していません。

- 影響 : AES-GCM を使用して 16,000 バイトを超えるデータを暗号化することができません。
- 回避策: AES-CBC などの代替メカニズムを使用するか、データを複数部分に分割して各部分を別々に暗号化できます。データを分割する場合、分割された暗号化テキストとその復号化を管理する必要があります。FIPS では AES-GCM の初期化ベクター (IV) を HSM で生成する必要があるため、AES-GCM で暗号化された部分的データはそれぞれ IV が異なります。
- 解決策のステータス: SDK を修正し、データバッファが大きすぎる場合は明示的に失敗するようにします。マルチパート暗号化を使用しなくても大きいバッファをサポートできる代替方法を評価しています。更新は AWS CloudHSM フォーラムとバージョン履歴ページで告知されます。

問題: 楕円曲線ディフィーヘルマン (ECDH) キーの導出が、HSM 内で部分的に実行されます

EC プライベートキーは常に HSM にありますが、キーの取得手順は複数のステップで実行されます。その結果、各ステップの中間結果がクライアントに存在します。ECDH キー導出サンプルは Java コードサンプルで入手できます。

- 影響: クライアント SDK 3 により、JCE に ECDH 機能が追加されました。`KeyAgreement` クラスを使用して SecretKey を取得すると、最初にクライアントで使用可能になり、HSM にインポートされます。その後、キーのハンドルがアプリケーションに返されます。
- 回避策: で SSL/TLS オフロードを実装している場合 AWS CloudHSM、この制限は問題ではない可能性があります。アプリケーションでキーが常に FIPS 境界内に留まる必要がある場合、ECDH キー取得に依存しない代替プロトコルの使用を検討してください。
- 解決策のステータス: SDK 5.16 は、ECDH とキー導出をサポートしており、これらの処理はすべて HSM 内で実行されます。

問題: KeyGenerator と KeyAttribute がキーサイズパラメータをビット数ではなくバイト数として誤って解釈します

KeyGenerator クラスの `init` 関数または AWS CloudHSM KeyAttribute 列挙値の `SIZE` 属性を使用してキーを生成する場合、API は引数がキービット数であるべきところ、キーバイト数であると誤って想定します。

- 影響: クライアント SDK バージョン 5.4.0～5.4.2 では、指定した API にキーサイズがバイトとして提供されることを誤って想定しています。
- 回避策: KeyGenerator クラスまたは KeyAttribute 列挙を使用する前にキーサイズをビットからバイトに変換し、クライアント SDK バージョン 5.4.0 から 5.4.2 を使用している場合は AWS CloudHSM JCE プロバイダーを使用してキーを生成します。
- 解決策のステータス: クライアント SDK バージョンを 5.5.0 以降にアップグレードしてください。これには、KeyGenerator クラスまたは KeyAttribute 列挙値を使用してキーを生成するときに、キーサイズがビット単位であることを正しく想定する修正が含まれています。

問題: クライアント SDK 5 から「不正な反射アクセスオペレーションが発生しました」という警告が表示されます

Java 11 でクライアント SDK 5 を使用すると、CloudHSM から次の Java 警告が表示されます。

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by
 com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore (file:/opt/cloudhsm/java/
cloudhsm-jce-5.6.0.jar) to field java.security .KeyStore.keyStoreSpi
WARNING: Please consider reporting this to the maintainers of
 com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
 access operations
WARNING: All illegal access operations will be denied in a future release
```

この問題は、クライアント SDK バージョン 5.8 以降で修正されています。

> **Note**
>
> OpenJDK 11 はサポートされなくなりました。クライアント SDK 5.17.1 は、OpenJDK 11 をサポートする最後のリリースです。

問題: JCE セッションプールが使い果たされています

影響: 次のメッセージが表示されると、JCE で操作を実行できなくなる可能性があります。

```
com.amazonaws.cloudhsm.jce.jni.exception.InternalException: There are too many
 operations
happening at the same time: Reached max number of sessions in session pool: 1000
```

回避方法:

- 影響が出ている場合は、JCE アプリケーションを再起動してください。
- オペレーションを実行する場合、オペレーションへの参照が失われる前に JCE オペレーションを完了する必要がある場合があります。

> **Note**
>
> オペレーションによっては、完了方法が必要な場合があります。

運用	完了方法
暗号	暗号化モードまたは復号モードの `doFinal()` ラップモードの `wrap()` アンラップモードの `unwrap()`
KeyAgreement	`generateSecret()` 、または `generateSecret(String)`

運用	完了方法
KeyPairGenerator	`generateKeyPair()` , `genKeyPair()` , または `reset()`
KeyStore	メソッド不要
MAC	`doFinal()` 、または `reset()`
MessageDigest	`digest()`、または `reset()`
SecretKeyFactory	メソッド不要
SecureRandom	メソッド不要
Signature	サインモードの `sign()` 検証モードの `verify()`

解決策のステータス: この問題は、クライアント SDK 5.9.0 以降で解決に向けて積極的に取り組んでいます。この問題を解決するには、クライアント SDK を以下のバージョンのいずれかにアップグレードしてください。

問題: getKey 操作によるクライアント SDK 5 メモリリーク

- 影響: API `getKey` オペレーションには、クライアント SDK バージョン 5.10.0 以前の JCE でメモリリークがあります。アプリケーションで `getKey` API を複数回使用している場合、メモリの増加につながり、その結果、アプリケーションのメモリフットプリントが増加します。時間が経つと、スロットリングエラーが発生したり、アプリケーションの再起動が必要になる場合があります。
- 回避策: クライアント SDK 5.11.0 にアップグレードすることをお勧めします。アップグレードができない場合は、アプリケーションで `getKey` API を複数回呼び出さないことをお勧めします。代わりに、以前の `getKey` オペレーションから以前に返されたキーを可能な限り再利用します。
- 解決策のステータス: クライアント SDK バージョンを 5.11.0 以降にアップグレードします。これには、この問題の修正が含まれています。

の OpenSSL Dynamic Engine の既知の問題 AWS CloudHSM

AWS CloudHSM用の OpenSSL Dynamic Engine SDK の既知の問題があります

トピック

- 問題: RHEL 6 および CentOS6 に AWS CloudHSM OpenSSL Dynamic Engine をインストールできない
- [問題] デフォルトでは、HSM への RSA オフロードのみがサポートされています
- 問題: HSM でキーを使用した OAEP パディングによる RSA 暗号化および復号化がサポートされていません。
- [Issue: (問題)] RSA のプライベートキー世代および ECC キーのみが HSM にオフロードされます。
- 問題: RHEL 8、CentOS 8、 Ubuntu 18.04 LTS にクライアント SDK 3 用の OpenSSL Dynamic Engine をインストールできない
- 問題: RHEL 9 (9.2 以降) での SHA-1 署名と検証の非推奨になっています。
- 問題: AWS CloudHSM OpenSSL Dynamic Engine が OpenSSL v3.x の FIPS プロバイダーと互換性がない
- 問題: SDK 5.16 以降、TLS 1.0 および TLS 1.1 の ECDSA 暗号スイートで SSL/TLS オフロードが失敗する

問題: RHEL 6 および CentOS6 に AWS CloudHSM OpenSSL Dynamic Engine をインストールできない

- 影響: OpenSSL Dynamic Engineは OpenSSL 1.0.2[f+] のみをサポートしています。デフォルトでは、RHEL 6 と CentOS 6 には OpenSSL 1.0.1 が付属します。
- 回避方法: RHEL 6 および CentOS 6 の OpenSSL ライブラリをバージョン 1.0.2[f+] にアップグレードします。

[問題] デフォルトでは、HSM への RSA オフロードのみがサポートされています

- [影響]: パフォーマンスを最大限に高めるために、SDK は乱数生成や EC-DH オペレーションなどの追加機能をオフロードするように構成されていません。
- [Workaround (回避策)]: 追加のオペレーションをオフロードする必要がある場合は、サポートケースを通じてお問い合わせください。

- [Resolution status (解決策のステータス)]: オフロードオプションを設定ファイルで設定するための SDK へのサポートを追加しています。更新は、利用可能なバージョン履歴ページで告知されます。

問題: HSM でキーを使用した OAEP パディングによる RSA 暗号化および復号化がサポートされていません。

- 影響: OAEP パディングによる RSA 暗号化および復号化に対するすべての呼び出しが、ゼロ除算エラーにより失敗します。これは、OpenSSL 動的エンジンがオペレーションを HSM にオフロードせずにフェイク PEM ファイルを使用してオペレーションをローカルで呼び出すために発生します。
- 解決策: AWS CloudHSM クライアント SDK 5 用の PKCS #11 ライブラリ または AWS CloudHSM クライアント SDK 5 の JCE プロバイダー を使用してこの手順を実行できます。
- 解決策のステータス: このオペレーションを正しくオフロードする SDK のサポートを追加する予定です。更新は、利用可能なバージョン履歴ページで告知されます。

[Issue: (問題)] RSA のプライベートキー世代および ECC キーのみが HSM にオフロードされます。

他のキータイプでは、OpenSSL AWS CloudHSM エンジンは通話処理には使用されません。代わりに、ローカルの OpenSSL エンジンが使用されます。これによって、ソフトウェアでローカルにキーが生成されます。

- [影響 :] フェイルオーバーがサイレントのため、HSM で安全に生成されたキーを受信していないことが確認できません。キーがソフトウェアで OpenSSL によってローカルで生成された場合、文字列 "`.....++++++`" を含む出力トレースが表示されます。オペレーションが HSM にオフロードされた場合には、このトレースは存在しません。キーが生成されていない、あるいは HSM に保存されていないため、キーを今後使用することはできません。]
- [Workaround: (回避方法)] OpenSSL エンジンがサポートするキータイプのみを使用します。その他のすべてのキータイプには、PKCS#11 あるいは JCE をアプリケーションで使用するか、または `key_mgmt_util` を CLI で使用します。

問題: RHEL 8、CentOS 8、 Ubuntu 18.04 LTS にクライアント SDK 3 用の OpenSSL Dynamic Engine をインストールできない

- 影響: デフォルトでは、RHEL 8、CentOS 8、Ubuntu 18.04 LTSは、クライアント SDK 3 用 OpenSSL Dynamic Engine と互換性がないバージョンを出荷しています。
- 防止策: OpenSSL 動的エンジン 対応の Linux プラットフォームを使用してください。対応プラットフォームの詳細は、対応プラットフォーム を参照してください。
- 解決ステータス: AWS CloudHSM は、クライアント SDK 5 用の OpenSSL Dynamic Engine でこれらのプラットフォームをサポートします。詳細については、対応プラットフォーム および OpenSSL Dynamic Engine を参照してください。

問題: RHEL 9 (9.2 以降) での SHA-1 署名と検証の非推奨になっています。

- 影響: 暗号化目的での SHA-1 メッセージ ダイジェストの使用は、RHEL 9 (9.2 以降) で非推奨になりました。その結果、OpenSSL Dynamic Engine を使用した SHA-1 による署名と検証オペレーションは失敗します。
- 回避策: シナリオで既存またはサードパーティーの暗号化署名の署名/検証に SHA-1 を使用する必要がある場合は、「nhancing RHEL Security: Understanding SHA-1 deprecation on RHEL 9 (9.2 以降)」 および「RHEL 9 (9.2+) Release Notes」を参照してください。

問題: AWS CloudHSM OpenSSL Dynamic Engine が OpenSSL v3.x の FIPS プロバイダーと互換性がない

- 影響: FIPS プロバイダーが AWS CloudHSM OpenSSL バージョン 3.x で有効になっているときに OpenSSL Dynamic Engine を使用しようとすると、エラーが発生します。
- 回避策: AWS CloudHSM OpenSSL バージョン 3.x で OpenSSL Dynamic Engine を使用するには、「デフォルト」プロバイダーが設定されていることを確認します。OpenSSL ウェブサイト でデフォルトのプロバイダーの詳細をご覧ください。

問題: SDK 5.16 以降、TLS 1.0 および TLS 1.1 の ECDSA 暗号スイートで SSL/TLS オフロードが失敗する

- 影響: TLS 1.0 および TLS 1.1 では署名に SHA-1 が使用されますが、これは FIPS 186-5 の要件を満たさないため、これらのバージョンで接続を試みると失敗します。

- 回避策: TLS バージョンをすぐにアップグレードできない場合は、ハッシュ強度要件を適用しない非 FIPS クラスターへ移行することができます。ただし、FIPS 準拠およびセキュリティのベストプラクティスの観点からは、TLS 1.2 または TLS 1.3 へのアップグレードをお勧めします。
- 解決策: 実装を TLS 1.2 または TLS 1.3 を使用するようにアップグレードしてください。TLS 1.0 と TLS 1.1 は、セキュリティ上の懸念から IETF (Internet Engineering Task Force) によって非推奨となっています。

のキーストレージプロバイダー (KSP) の既知の問題 AWS CloudHSM

以下は、 AWS CloudHSM用キーストレージプロバイダー (KSP) の既知の問題です。

トピック

- 問題: 証明書ストアの検証が失敗する
- 問題: クライアント SDK 5 の SDK3 互換モード使用時に、証明書ストア内のコンテナ名に不整合が発生する

問題: 証明書ストアの検証が失敗する

クライアント SDK バージョン 5.14 および 5.15 を使用している場合に `certutil -store my CERTIFICATE_SERIAL_NUMBER` を呼び出すと次のエラーがスローされます。

```
ERROR: Could not verify certificate public key against private key
```

- 影響: クライアント SDK 5 で作成された証明書ストアは、`certutil` を使って検証することができません。
- 回避策: プライベートキーを使用してファイルに署名し、パブリックキーを使用して署名を検証することで、証明書に関連付けられたキーペアを検証します。これは、Microsoft SignTool を使用し、ここに記載されている手順に従うことで実行できます。
- 解決策のステータス: `certutil` を使用した証明書検証をサポートする修正に取り組んでいます。修正は、利用可能なバージョン履歴ページで告知されます。

問題: クライアント SDK 5 の SDK3 互換モード使用時に、証明書ストア内のコンテナ名に不整合が発生する

`certutil -store my CERTIFICATE_SERIAL_NUMBER` コマンドを使用して、 AWS CLI 5.16.0 の generate-file コマンドを使用してキー参照ファイルが生成された証明書を表示すると、次のエラーが発生します。

```
ERROR: Container name inconsistent: CONTAINER_NAME
```

このエラーは、証明書に保存されているコンテナ名 と、CloudHSM CLI によって生成されたキー参照ファイル名が一致しないことが原因で発生します。

- 影響: このエラーが発生していても、証明書および関連付けられたキーは完全に機能します。これらの証明書を利用するアプリケーションは、問題なく通常どおり動作し続けます。
- 回避策: このエラーを解消するには、キー参照ファイル名を Simple または Unique のコンテナ名に変更してください。次に示す `certutil -store my` コマンドのサンプル出力を参照してください。

  ```
  Subject: CN=www.website.com, OU=Organizational-Unit, O=Organization, L=City, S=State,
   C=US
  Non-root Certificate
  Cert Hash(sha1): 1add52
  Key Container = 7e3c-b2f5
  Simple container name: tq-3daacd89
  Unique container name: tq-3daacd89
  ERROR: Container name inconsistent: 7e3c-b2f5
  ```

  デフォルトでは、キー参照ファイルは `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition` に保存されます。

  1. キー参照ファイルの名前をシンプルなコンテナ名に変更します。
  2. その後、新しいキーコンテナ名を使用して証明書ストアを修復します。詳細については、KSP の 移行のステップ 12～14 を参照してください。
- 解決策のステータス: この問題は クライアント SDK バージョン 5.16.1 で修正されています。問題を解決するには、クライアント SDK を 5.16.1 以降 にアップグレードしてください。

の OpenSSL プロバイダーの既知の問題 AWS CloudHSM

これらは、 の OpenSSL プロバイダーの既知の問題です AWS CloudHSM。

トピック

- 問題: OpenSSL プロバイダーで使用する場合の OpenSSL CLI のエラー
- 問題: FIPS クラスターで Ed25519 TLS オフロードがサポートされていない
- 問題: OpenSSL が 3.5 より前のプラットフォームで ML-DSA オペレーションが失敗する
- 問題: Amazon Linux 2023 と RHEL で「共有署名アルゴリズムなし」で ML-DSA TLS ハンドシェイクが失敗する

問題: OpenSSL プロバイダーで使用する場合の OpenSSL CLI のエラー

- 影響: AWS CloudHSM OpenSSL プロバイダーは OpenSSL CLI オペレーション (CSR 作成、証明書署名) をサポートしていませんでした。CSR および証明書オペレーションには OpenSSL Dynamic Engine を使用する必要があります。
- 解決ステータス: クライアント SDK 5.18.0 はこの問題を解決します。OpenSSL プロバイダーは、すべてのキータイプ (RSA、EC、Ed25519、ML-DSA) の OpenSSL CLI オペレーションをサポートするようになりました。修正を活用するには、バージョン 5.18.0 以降にアップグレードします。

問題: FIPS クラスターで Ed25519 TLS オフロードがサポートされていない

- 影響: Ed25519 キータイプは FIPS モードクラスターでは使用できません。FIPS クラスターでの TLS オフロードにこのキータイプを使用しようとすると失敗します。
- 解決策: Ed25519 は、FIPS 以外のクラスターでのみ使用します。FIPS クラスターの場合は、TLS オフロードに RSA または EC キータイプを使用します。

問題: OpenSSL が 3.5 より前のプラットフォームで ML-DSA オペレーションが失敗する

- 影響: ML-DSA キータイプ (ML-DSA-44、ML-DSA-65、ML-DSA-87) では、CSR の作成、証明書の作成、TLS オフロードに OpenSSL 3.5 以降が必要です。古いシステム OpenSSL を使用するプラットフォームでは、ML-DSA オペレーションは「サポートされていないアルゴリズム」エラーで失敗します。

- 解決策: OpenSSL 3.5 以降でプラットフォームを使用するか、ML-DSA オペレーション用のカスタム OpenSSL 3.5+ バイナリを構築します。

問題: Amazon Linux 2023 と RHEL で「共有署名アルゴリズムなし」で ML-DSA TLS ハンドシェイクが失敗する

- 影響: ML-DSA 証明書を使用する TLS 接続は、Amazon Linux 2023、RHEL 9、および RHEL 10 プラットフォームでエラー で失敗しますtls1_set_server_sigalgs:no shared signature algorithms。これは、システム全体の crypto-policies フレームワークに、デフォルトのSignatureAlgorithms許可リストに ML-DSA 署名アルゴリズム (mldsa44、mldsa65、mldsa87) が含まれていないために発生します。TLS 以外のオペレーション (キーの生成、署名、検証) は影響を受けません。Ubuntu 26.04 LTS は crypto-policies フレームワークを使用しないため、影響を受けません。
- 解決策: プラットフォームでポスト量子 (PQ) 暗号化サブポリシーを有効にします。
  - Amazon Linux 2023 (AL2023.12 以降が必要): を実行しますsudo update-crypto-policies --set DEFAULT:PQ。詳細については、「Amazon Linux 2023 ユーザーガイド」の「ポスト量子暗号化ポリシー」を参照してください。
  - RHEL 9 (RHEL 9.8 以降が必要): を実行しsudo dnf update crypto-policies、次に を実行しますsudo update-crypto-policies --set DEFAULT:PQ。詳細については、RHEL 9 ドキュメントの「システム全体の暗号化ポリシーの使用」を参照してください。
  - RHEL 10 (RHEL 10.1 以降が必要): ML-DSA はDEFAULTポリシーで自動的に有効になります。を実行してsudo dnf update、最新の crypto-policies パッケージがあることを確認します。詳細については、RHEL 10 ドキュメントの「システム全体の暗号化ポリシーの使用」を参照してください。

の CloudHSM CLI の既知の問題 AWS CloudHSM

以下の問題は CloudHSM CLI に影響します AWS CloudHSM。

トピック

- key-reference フィルターがセッションキーの選択に失敗する

`key-reference` フィルターがセッションキーの選択に失敗する

を使用してセッション (エフェメラル) キーkey-referenceをフィルタリングするコマンドは、 エラー で失敗しますUX000: Ephemeral key is not expected because we cannot build it without HSM Connection。

key set-attribute コマンドは影響を受けず、 によってセッションキーを選択できますkey-reference。

- 回避策: 属性ベースのフィルター (などattr.label) を使用してセッションキーを選択します。複数のセッションキーが同じ属性を共有する場合は、key-referenceフィルターでキーセット属性を使用して最初に一意のラベルを割り当て、次にラベルでフィルタリングします。
- 解決ステータス: この問題はクライアント SDK 5.18.0 で解決されています。key-reference フィルターは、CloudHSM CLI と JCE でセッション (エフェメラル) キーを選択するようになりました。修正を活用するには、バージョン 5.18.0 以降にアップグレードします。

で Amazon Linux 2 を実行する Amazon EC2 インスタンスの既知の問題 AWS CloudHSM

以下の問題は、Amazon Linux 2 で実行されている AWS CloudHSM および Amazon EC2 インスタンスに影響します。 Amazon EC2

問題: Amazon Linux 2 バージョン 2018.07 は、現在 AWS CloudHSM SDKs と互換性のない更新された`ncurses`パッケージ (バージョン 6) を使用しています

AWS CloudHSM cloudhsm_mgmt_util または key_mgmt_util を実行すると、次のエラーが返されます。

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:
 libncurses.so.5: cannot open shared object file: No such file or directory
```

- 影響: Amazon Linux 2 バージョン 2018.07 で実行されているインスタンスは、すべての AWS CloudHSM ユーティリティを使用することはできません。
- 防止策 : Amazon Linux 2 EC2 インスタンスで次のコマンドを発行して、対応している ncurses パッケージ (バージョン 5) をインストールします。

```
sudo yum update && yum install ncurses-compat-libs
```

- 解決策のステータス: この問題は、 AWS CloudHSM クライアント 1.1.2 のリリースで解決されています。修正のメリットを享受するには、このクライアントにアップグレードする必要があります。

サードパーティーアプリケーションを と統合する際の既知の問題 AWS CloudHSM

以下の問題は、サードパーティーのアプリケーションと統合 AWS CloudHSM する場合に影響します。

問題: クライアント SDK 3 で、マスターキーの生成時に Oracle が設定する PKCS #11 属性 **CKA_MODIFIABLE** がサポートされていません

この制限は PKCS #11 ライブラリで定義されています。詳細については、「サポートされている PKCS #11 属性」の注釈 1 を参照してください。

- 影響 : Oracle マスターキーの作成に失敗する。
- 回避方法 : 新しいマスターキーを作成するときに、特別な環境変数 `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` を TRUE に設定します。この環境変数は、マスターキーの生成にのみ必要であり、この環境変数を他のものに使用する必要はありません。たとえば、作成した最初のマスターキーにこの変数を使用し、マスターキーの工ディションの口ーテーションを行う場合にのみ、この環境変数を再度使用します。詳細については、「Oracle TDE マスター暗号化キーの生成」を参照してください。
- 解決策のステータス : HSM ファームウェアを改善して、CKA_MODIFABLE 属性を完全にサポートしています。更新は、 AWS CloudHSM フォーラムとバージョン履歴ページで発表されます。

AWS CloudHSM クラスターの変更に関する既知の問題

クラスターの HSM タイプを変更するために modify-cluster API を使用しようとするお客様に、以下の問題が影響します。

トピック

- 問題: PBKDF2 の反復回数の増加によるログインレイテンシーの増加
- 問題: トークンキー作成により HSM タイプを変更できない

問題: PBKDF2 の反復回数の増加によるログインレイテンシーの増加

- 影響: ユーザー数が多いクラスターでは、移行期間が大幅に長くなる可能性があります。これは、hsm1.medium のバックアップを hsm2m.medium に初めて復元する際、バックアップ復元プロセスがユーザーごとに PBKDF2 処理を実行するように変更されたためです。
- 解決ステータス: 2025 年 12 月 20 日より前に新しい hsm2m.medium インスタンスを作成した場合、または hsm1.medium から hsm2m.medium に移行した場合は、パスワードをリセットして、ログインオペレーション用に実装されたパフォーマンスの向上を活用する必要があります。手順については、変更パスワードを参照してください。

問題: トークンキー作成により HSM タイプを変更できない

- 影響: トークンキーを使用するワークロードを実行しているお客様は、移行を開始できません。これは、HSM タイプの変更中にデータ損失を防ぐため、HSM が書き込み制限モードに移行するためです。
- 回避策: トークンキーの作成および削除を停止し、7 日間待機してください。また、以下のいずれかに該当する場合は、サポートに連絡してください。
  - トークンキー移行のブロックに対応できず、ブルー/グリーンデプロイも行えない。
  - 移行期間中のトークンキー操作のブロックには対応できるが、7 日間待てない。
- 解決策のステータス: この問題は解決済みです。トークンキーを使用するワークロードのお客様でも移行を開始できるようになりました。移行期間中は、トークンキーの作成と削除がブロックされます。

hsm2.medium での AWS CloudHSM クライアントバージョン 5.12.0 を使用したオペレーション失敗の既知の問題

AWS CloudHSM クライアントバージョン 5.12.0 を使用する場合 AWS CloudHSM の以下の問題の影響

問題: get-attribute オペレーション中のエラー

hsm1.medium から hsm2m.medium へ移行しており、CloudHSM クライアント SDK 5.12.0 を使用している場合、属性処理に関するエラーが発生する可能性があります。

クライアントログには、次のエラーメッセージが表示される場合があります: `Error in deserialization of data: Invalid integer conversion`

影響: クライアント バージョン 5.12.0 を使用していると、以下のオペレーションが失敗します

- PKCS#11 SDK では、C_GetAttributeValue 呼び出しが失敗します
- CloudHSM CLI では、key list コマンドの出力に属性が表示されません
- CloudHSM CLI では、hsm1.medium で生成したキーに対して key generate-file が失敗する場合があります

解決策: この問題は最新の SDK バージョンで修正されています。最新バージョンへのアップグレードをお勧めします。

AWS CloudHSM クライアント SDK 3 キー同期の失敗

クライアント SDK 3 では、クライアント側の同期が失敗した場合、 はベストエフォートレスポンス AWS CloudHSM を行い、作成された (および不要になった) 不要なキーをクリーンアップします。このプロセスでは、不要なキーマテリアルを直ちに除去すること、あるいは後で除去するために不要な材料をマーキングすることを含んでいます。どちらの場合も、解決するためにはお客様からのアクションは必要ありません。が不要なキーマテリアルを削除 AWS CloudHSM できず、マークできないというまれなケースでは、キーマテリアルを削除する必要があります。

問題: トークンキーの生成、インポート、またはアンラップ操作を試行し、tombstone への失敗を指定するエラーが表示される。

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

原因: AWS CloudHSM 不要なキーマテリアルの削除とマークに失敗しました。

解決方法: クラスター内の HSM には、不要とマークされていない不要なキーマテリアルが含まれています。キーマテリアルを手動で削除する必要があります。不要なキーマテリアルを手動で削除するには、[key_mgmt_util (KMU)]、または[PKCS #11] あるいは JCE プロバイダーからの API を使用します。詳細については、deleteKey または クライアント SDK を参照してください。

トークンキーの耐久性を高めるために、 は、クライアント側の同期設定で指定された最小数の HSMs で成功しないキー作成オペレーションを AWS CloudHSM 失敗させます。詳細については、Key Synchronization in AWS CloudHSM を参照してください。

AWS CloudHSM クライアント SDK 3 が pkpspeed ツールで HSM のパフォーマンスを検証する

このトピックでは、クライアント SDK 3 で AWS CloudHSM ハードウェアセキュリティモジュール (HSM) のパフォーマンスを検証する方法について説明します。

AWS CloudHSM クラスター内の HSMs のパフォーマンスを確認するには、クライアント SDK 3 に含まれている pkpspeed (Linux) または pkpspeed_blocking (Windows) ツールを使用できます。pkpspeed ツールは理想的な条件下で実行され、PKCS11 などの SDK を経由せずに HSM を直接呼び出して操作を実行します。スケーリングのニーズを判断するために、アプリケーションの負荷を個別にテストすることを推奨します。ランダム (I)、ModeXP (R)、EC ポイント mul (Y) の各テストの実施は推奨しません。

Linux EC2 インスタンスにクライアントをインストールする方法の詳細については、「CMU 用の AWS CloudHSM クライアントをインストールして設定する (Linux)」を参照してください。Windows インスタンスにクライアントをインストールする方法の詳細については、「CMU 用の AWS CloudHSM クライアントをインストールして設定する (Windows)」を参照してください。

AWS CloudHSM クライアントをインストールして設定したら、次のコマンドを実行してクライアントを起動します。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合。

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

クライアントソフトウェアをインストール済みの場合は、必要に応じて最新バージョンの pkpspeed をダウンロードしてインストールします。pkpspeed ツールは、Linux の `/opt/cloudhsm/bin/pkpspeed` または Windows の `C:\Program Files\Amazon\CloudHSM\` にあります。

pkpspeed を使用するには、pkpspeed コマンドあるいは pkpspeed_blocking.exe を実行し、HSM の Crypto User (CU) のユーザー名とパスワードを指定します。次に、以下の推奨事項を考慮に入れながら、使用するオプションを設定します。

レコメンデーションをテストする

- RSA の署名および検証オペレーションのパフォーマンスをテストするには、Linux または Windows のオプション B の RSA_CRT 暗号を選択します。RSA は選択しないでください (option A

in Windows のオプション A)。暗号は同じですが、RSA_CRT はパフォーマンス用に最適化されています。

- 少数のスレッドで開始します。AES パフォーマンスをテストする場合、通常、1 つのスレッドで十分に最大のパフォーマンスが示されます。RSA パフォーマンスをテストする場合 (RSA_CRT) は、通常、3〜4 つのスレッドで十分です。

pkpspeed ツールの設定可能なオプション

- FIPS モード: AWS CloudHSM は常に FIPS モードです (詳細については、AWS CloudHSM FAQs」を参照してください) 。これは、 AWS CloudHSM 「 ユーザーガイド」に記載されている CLI ツールを使用し、FIPS モードのステータスを示す CMU を使用して AWS CloudHSM クラスター内の各 HSM のハードウェア情報を取得する コマンドを実行することで確認できます。
- テストタイプ (ブロッキングとノンブロッキング): スレッド方式でのオペレーションの実行方法を指定します。ノンブロッキングを使用すると良い数値が得られる可能性が高くなるでしょう。これは、スレッドと同時実行性を利用するためです。
- スレッド数: テストを実行するスレッドの数。
- テスト実行時間 (最大 = 600): pkpspeed は「オペレーション/秒」で測定された結果を生成し、テストが実行される毎秒この値を報告します。例えば、テストを 5 秒間実行する場合、出力は次のサンプル値のようになります。
  - `OPERATIONS/second 821/1`
  - `OPERATIONS/second 833/1`
  - `OPERATIONS/second 845/1`
  - `OPERATIONS/second 835/1`
  - `OPERATIONS/second 837/1`

pkpspeed ツールで実行できるテスト

- AES GCM: AES GCM モードの暗号化をテストします。
- ベーシック 3DES CBC: 3DES CBC モードの暗号化をテストします。今後の変更については、以下の注記「1」を参照してください。
- ベーシック AES: AES CBC/ECB 暗号化をテストします。
- ダイジェスト: ハッシュダイジェストをテストします。
- ECDSA サイン: ECDSA サインをテストします。

- ECDSA 検証: ECDSA 検証をテストします。
- FIPS ランダム: FIPS 準拠の乱数の生成をテストします (注: これはブロッキングモードでのみ使用できます)。
- HMAC: HMAC をテストします。
- ランダム: FIPS 140-2 HSM を使用しているため、このテストは関係ありません。
- RSA 非 CRT と RSA_CRT の比較: RSA サインと検証オペレーションをテストします。
- RSA OAEP 暗号: RSA OAEP の暗号化をテストします。
- RSA OAEP 復号: RSA OAEP の復号化をテストします。
- RSA プライベート復号化非 CRT: RSA プライベートキー暗号化 (非最適化) をテストします。
- RSA プライベート復号化 CRT: RSA プライベートキー暗号化 (最適化) をテストします。
- RSA PSS サイン: RSA PSS サインをテストします。
- RSA PSS 検証: RSA PSS 検証をテストします。
- RSA パブリックキー暗号: RSA パブリックキー暗号化をテストします。

RSA パブリックキー暗号化、RSA プライベート復号化非 CRT、RSA プライベートキー復号化 CRT でも、ユーザーに次の回答を求めるプロンプトが表示されます。

```
Do you want to use static key [y/n]
```

y が入力された場合、事前に計算されたキーが HSM にインポートされます。

n が入力された場合、新しいキーが生成されます。

[1] NIST ガイダンスに従い、2023 年以降の FIPS モードのクラスターでは、これは許可されません。FIPS 以外のモードのクラスターでは、2023 年以降も許可されます。詳細については、「FIPS 140 コンプライアンス: 2024 年 メカニズムの非推奨」を参照してください。

例

以下の例では、RSA オペレーションと AES オペレーションにおける HSM のパフォーマンスをテストするために pkpspeed (Linux) または pkpspeed_blocking (Windows) で選択できるオプションを示します。

Example- pkpspeed を使用した RSA パフォーマンスのテスト

ここの例は、Windows、Linux、および互換性のあるオペレーティングシステムで実行できます。

Linux

これらの手順は、Linux および互換性のあるオペレーティングシステムで使用してください。

```
/opt/cloudhsm/bin/pkpspeed -s CU user name -p password

SDK Version: 2.03

        Available Ciphers:
                AES_128
                AES_256
                3DES
                RSA  (non-CRT. modulus size can be 2048/3072)
                RSA_CRT (same as RSA)
For RSA, Exponent will be 65537

Current FIPS mode is: 00002
Enter the number of thread [1-10]: 3
Enter the cipher: RSA_CRT
Enter modulus length: 2048
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 245 bytes...
[Test duration is 60 seconds]

Do you want to use static key[y/n] (Make sure that KEK is available)?n
```

Windows

```
c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password

Please select the test you want to run

RSA non-CRT------------------->A
RSA CRT----------------------->B
Basic 3DES CBC---------------->C
Basic AES--------------------->D
FIPS Random------------------->H
Random------------------------>I
AES GCM ---------------------->K

eXit------------------------->X
B
```

```
Running 4 threads for 25 sec

Enter mod size(2048/3072):2048
Do you want to use Token key[y/n]n
Do you want to use static key[y/n] (Make sure that KEK is available)?  n
OPERATIONS/second                821/1
OPERATIONS/second                833/1
OPERATIONS/second                845/1
OPERATIONS/second                835/1
OPERATIONS/second                837/1
OPERATIONS/second                836/1
OPERATIONS/second                837/1
OPERATIONS/second                849/1
OPERATIONS/second                841/1
OPERATIONS/second                856/1
OPERATIONS/second                841/1
OPERATIONS/second                847/1
OPERATIONS/second                838/1
OPERATIONS/second                843/1
OPERATIONS/second                852/1
OPERATIONS/second                837/
```

Example- pkpspeed を使用した AES パフォーマンスのテスト

Linux

これらの手順は、Linux および互換性のあるオペレーティングシステムで使用してください。

```
/opt/cloudhsm/bin/pkpspeed -s <CU user name> -p <password>

SDK Version: 2.03

        Available Ciphers:
                AES_128
                AES_256
                3DES
                RSA  (non-CRT. modulus size can be 2048/3072)
                RSA_CRT (same as RSA)
For RSA, Exponent will be 65537

Current FIPS mode is: 00000002
Enter the number of thread [1-10]: 1
Enter the cipher: AES_256
```

```
Enter the data size [1-16200]: 8192
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 8192 bytes...
```

Windows

```
c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password
login as USER
Initializing Cfm2 library
        SDK Version: 2.03

 Current FIPS mode is: 00000002
Please enter the number of threads [MAX=400] : 1
Please enter the time in seconds to run the test [MAX=600]: 20


Please select the test you want to run

RSA non-CRT------------------->A
RSA CRT----------------------->B
Basic 3DES CBC---------------->C
Basic AES--------------------->D
FIPS Random------------------->H
Random------------------------>I
AES GCM ---------------------->K

eXit------------------------->X
D

Running 1 threads for 20 sec

Enter the key size(128/192/256):256
Enter the size of the packet in bytes[1-16200]:8192
OPERATIONS/second                9/1
OPERATIONS/second                10/1
OPERATIONS/second                11/1
OPERATIONS/second                10/1
OPERATIONS/second                10/1
OPERATIONS/second                10/...
```

AWS CloudHSM クライアント SDK 5 ユーザーまたはポリシーに整合性のない値が含まれている

AWS CloudHSM は、クラスター内の HSMs 間でユーザーまたはポリシー (mTLS 設定など) を自動的に同期しません。CloudHSM CLI はHSMs 間でこれらのオペレーションのベストエフォート同期を実行しますが、不整合が発生する可能性があります。このページでは、ユーザーとポリシーの両方の不整合を特定して解決する方法について説明します。

一貫性のないユーザー値を解決する

AWS CloudHSM Client SDK 5 の `user list` コマンドは、クラスター内のすべてのユーザーとユーザープロパティのリストを返します。ユーザーのプロパティのいずれかに「不整合」という値がある場合、このユーザーはクラスター間で同期されません。つまり、そのユーザーはクラスター内の異なる HSM に異なるプロパティで存在することになります。どのプロパティに一貫性がないかによって、異なる修復手順を取ることができます。

以下の表には、1 人のユーザーの不整合を解決する手順が記載されています。1 人のユーザーに複数の不整合がある場合は、以下の手順を上から順に実行して解決してください。不整合があるユーザーが複数いる場合は、ユーザーごとにこのリストを確認し、そのユーザーの不整合を完全に解決してから次のユーザーに進みます。

> **Note**
>
> これらの手順を実行するには、管理者として ログインするのが理想的です。管理者アカウントに一貫性がない場合は、管理者にログインし、すべてのプロパティが一致するまで手順を繰り返してください。管理者アカウントの一貫性が保たれたら、その管理者を使用してクラスター内の他のユーザーを同期できます。

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法
ユーザーの「役割」に「一貫性がない」	`{ "username": "test_user",`	このユーザーは、一部の HSM では CryptoUser であり、他の HSM では管理者です。これは、2 つの	1. 管理者として ログインします。 2. すべての HSM のユーザーを削除します。

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法
	"role": "**inconsistent** ", "locked": "false", "mfa": [], "cluster-coverage": "full" }	SDK が同じユーザーを、異なるロールで同時に作成しようとした場合に発生する可能性があります。このユーザーを削除し、目的のロールで再作成する必要があります。	user delete --username **<user's name>** --role admin user delete --username **<user's name>** --role crypto-user 3. 目的のロールを持つユーザーを作成します。 user create --username **<user's name>** --role **<desired role>**

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法

ユーザーの「クラスターカバレッジ」は「一貫性がない」	<pre>{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [],
	"cluster-coverage":
	"**inconsistent** "
	}</pre>
		このユーザーはクラスター内の HSM のサブセットに存在します。これは、user create が部分的に成功した場合や、user delete が部分的に成功した場合に発生する可能性があります。
		
		このユーザーをクラスターから作成または削除して、前のオペレーションを完了する必要があります。
			ユーザーが存在しないはずの場合は、以下の手順に従ってください。
			
			1. 管理者としてログインします。
			2. 次のコマンドを実行します。
			
			user delete --username**<user's name>** --role admin
			3. 次のコマンドを実行します。
			
			user delete --username**<user's name>** --role crypto-user
			
			そのユーザーが存在しているはずなら、以下の手順に従ってください。
			
			1. 管理者としてログインします。
			2. 次のコマンドを実行します。
			
			user create --username **<user's name>**

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法
			--role **<desired role>**

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法

ユーザーの「ロック済み」パラメータが「不整合」または「true」です	<pre>{
	"username": "test_user",
	"role": "crypto-user",
	"locked": inconsistent,
	
	"mfa": [],
	"cluster-coverage": "full"
	}</pre>
		このユーザーは HSM のサブセットでロックアウトされています。
		
		これは、ユーザーが間違ったパスワードを使用し、クラスター内の HSM のサブセットにのみ接続した場合に発生する可能性があります。
		
		クラスター全体で一貫性を持たせるには、ユーザーの認証情報を変更する必要があります。
			ユーザーが MFA を有効にしている場合は、以下の手順で行います。
			
			1. 管理者としてログインします。
			2. 次のコマンドを実行して、MFA を一時的に無効にします。
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			
			3. すべての HSM にログインできるようにユーザーのパスワードを変更します。
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			ユーザーの MFA を有効にする必要がある場合は、以下の手順で行います。

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法
			1. ユーザーにログインして MFA を再度有効にしてもらいます (その場合、ユーザーはトークンに署名し、パブリックキーを PEM ファイルに提供する必要があります)。 user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法

MFA のステータスが「不整合」	<pre>{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [
	{
	"strategy":
	"token-sign",
	"status":
	"**inconsistent** "
	}
	],
	"cluster-coverage":
	"full"
	}</pre>
		このユーザーは、クラスター内の HSM ごとに異なる MFA フラグを持っています。
		
		これは、MFA オペレーションが HSM のサブセットでのみ完了した場合に発生する可能性があります。
		
		ユーザーのパスワードをリセットし、MFA を再度有効にできるようにする必要があります。
			ユーザーが MFA を有効にしている場合は、以下の手順で行います。
			
			1. 管理者としてログインします。
			2. 次のコマンドを実行して、MFA を一時的に無効にします。
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			3. また、ユーザーがすべての HSM にログインできるように、ユーザーのパスワードを変更する必要があります。
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			ユーザーの MFA を有効にする必要がある

プロパティに一貫性がありません	ユーザーリストの出力例	影響	復旧方法
			場合は、以下の手順で行います。 1. ユーザーにログインして MFA を再度有効にしてもらいます (その場合、ユーザーはトークンに署名し、パブリックキーを PEM ファイルに提供する必要があります)。 user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

一貫性のない mTLS ポリシー値を解決する

ユーザーと同様に、mTLS ポリシー (信頼アンカーとエンフォースメントレベル) は HSMs 間で自動的に同期されません。CloudHSM CLI は、mTLS コマンドを実行するときにベストエフォート同期を実行しますが、不整合が発生する可能性があります。次のコマンドを使用して、mTLS 設定の同期ステータスを確認できます。

mTLS トラストアンカーの同期を確認する

cluster mtls list-trust-anchors コマンドを実行して、トラストアンカーの同期ステータスを確認します。出力では、各トラストアンカーに `cluster-coverage`フィールドがあります。値が「フルHSMs に存在します。値が「フルHSMs 間で同期されません。

```
{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```

トラストアンカーのクラスターカバレッジが一貫していない場合は、登録または登録解除コマンドを再実行してオペレーションを完了します。

- 一部の HSMs:

  cluster mtls register-trust-anchor --path **<path-to-certificate>**
- 削除する必要があるトラストアンカーの登録解除を完了するには:

  cluster mtls deregister-trust-anchor --certificate-reference **<certificate-reference>**

mTLS の設定の詳細については、「クライアントと 間の相互 TLS の設定 AWS CloudHSM」を参照してください。

AWS CloudHSM クライアント SDK 5 ユーザーレプリケートの失敗

CloudHSM CLI の `user replicate` コマンドは、クローンされた AWS CloudHSM クラスター間でユーザーをレプリケートします。このガイドでは、ソースクラスター内部、またはソースクラスターと宛先クラスター間の不整合によって発生する障害への対処方法を説明します。ユーザーレプリケートは、次の属性を確認することで、ユーザーが一貫しているかどうかを検証します。

- ユーザーロール
- アカウントロック状態
- クォーラム状態
- 多要素認証 (MFA) ステータス

問題: 選択したユーザーがクラスター全体で同期されない。

ユーザーレプリケーションプロセスでは、ソースクラスター全体でユーザーの同期状態を確認します。ユーザーの属性が "inconsistent" という値になっている場合、そのユーザーがクラスター内で同期されていないことを意味します。ユーザーレプリケーションは次のエラーメッセージとともに失敗します。

```
{
  "error_code": 1,
  "data": "Specified user is inconsistent across the cluster"
}
```

ソースクラスター内のユーザーの非同期を確認するには、次の手順を実行します。

- CloudHSM CLI で、`user list` コマンドを実行します。

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "example-inconsistent-user",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "inconsistent"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
```

```
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

解決策: ソースクラスター全体でユーザー属性を同期する

- ソースクラスター全体でユーザー情報を同期するには、AWS CloudHSM クライアント SDK 5 ユーザーまたはポリシーに整合性のない値が含まれている を参照してください。

問題: 宛先クラスターに、属性が異なるユーザーが存在する

宛先クラスター内の 1 つ以上の HSM に、同じ参照を持つユーザーが既に存在しており、そのユーザー属性が異なる場合、次のエラーが発生する可能性があります。

```
{
  "error_code": 1,
  "data": "User replicate failed on 1 of 3 connections"
}
```

解決策

1. どちらのユーザー情報を保持すべきか判断します。
2. 不要なユーザーを、該当クラスターで `user delete` コマンドを実行して削除します。詳細については「CloudHSM CLI で AWS CloudHSM ユーザーを削除する」を参照してください。
3. `user replicate` コマンドを実行して、ユーザーをレプリケートします。

問題: hsm2m.medium から hsm1.medium へのユーザーレプリケートが失敗する

hsm2m.medium から hsm1.medium へのユーザーレプリケートはサポートされていません。hsm2m.medium のソースクラスターから hsm1.medium の宛先クラスターへユーザーをレプリケートしようとすると、次のエラーが発生します。

```
{
  "error_code": 1,
  "data": "User replicate failed on 1 of 1 connections"
}
```

解決策

- CloudHSM CLI のユーザー管理機能を使用して、欠落しているユーザーを手動で再作成します。

AWS CloudHSM クライアント SDK 5 キーレプリケートの失敗

CloudHSM CLI の `key replicate` コマンドは、ソース AWS CloudHSM クラスターから宛先 AWS CloudHSM クラスターにキーをレプリケートします。このガイドでは、ソースクラスター内部、またはソースクラスターと宛先クラスター間の不整合によって発生する障害への対処方法を説明します。

問題: 選択したキーがクラスター全体で同期されない。

キーレプリケーションプロセスでは、ソースクラスター全体でキーの同期状態を確認します。キー情報またはキー属性のいずれかが、"inconsistent" という値になっている場合、そのキーがクラスター内で同期されていないことを意味します。キーレプリケーションは次のエラーメッセージとともに失敗します。

```
{
  "error_code": 1,
  "data": "The selected key is not synchronized throughout the cluster"
}
```

ソースクラスター内のキーの非同期を確認するには、次の手順を実行します。

1. CloudHSM CLI で、`key list` コマンドを実行します。
2. `--filter` フラグを使用してキーを指定します。
3. `--verbose` フラグを追加して、キーのカバレッジ情報を含む詳細な出力を表示します。

```
aws-cloudhsm > key list --filter attr.label=example-desynchronized-key-label --verbose
{
  "error_code": 0,
  "data": {
```

```
    "matched_keys": [
      {
        "key-reference": "0x000000000048000f",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "example-desynchronized-key-label",
          "id": "0x",
          "check-value": "0xbe79db",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": "inconsistent",
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 16
        }
      }
```

```
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

解決策: ソースクラスター全体でキー情報と属性を同期する

ソースクラスター全体でキー情報と属性を同期させるには、次の手順を実行します。

1. キー属性の不一致がある場合: 該当するキーに対して `key set-attribute` コマンドを使用し、目的の属性に設定します。
2. 共有ユーザー範囲の不一致がある場合: `key share` または `key unshare` コマンドを使用して、対象のユーザーとのキー共有状況を調整します。

問題: 同じリファレンスのキーが、宛先クラスターに存在するが、情報または属性が異なる

宛先クラスターに、同じキー参照を持つキーがすでに存在し、そのキー情報または属性が異なる場合、次のエラーが発生する可能性があります。

```
{
  "error_code": 1,
  "data": "Key replicate failed on 1 of 3 connections"
}
```

解決策

1. どちらのキー情報または属性を保持すべきか判断します。
2. 不要なキーのバージョンを、該当クラスターで `key delete` コマンドを使用して削除します。
3. 正しいバージョンを保持しているクラスターからキーをレプリケートします。

AWS CloudHSM キーの可用性チェック中にエラーが表示される

問題: AWS CloudHSM ハードウェアセキュリティモジュール (HSM) が次のエラーを返しています。

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

原因: キーの可用性チェックでは、まれではありますが、紛失する可能性のあるキーを探します。このエラーは通常、HSM が 1 つしかないクラスター、または HSM が 2 つあるクラスターで片方が交換されている間に発生します。このような状況では、以下の顧客オペレーションが原因で上記のエラーが発生した可能性があります。

- CloudHSM CLI の generate-symmetric カテゴリ または CloudHSM CLI の generate-asymmetric-pair カテゴリ のようなコマンドを使用して新しいキーが生成されました。
- CloudHSM CLI でユーザーのキーを一覧表示する オペレーションが開始されました。
- SDK の新しいインスタンスが開始されました。

> **Note**
>
> OpenSSL は SDK の新しいインスタンスを頻繁にフォークします。

解決策/推奨事項: このエラーの発生を防ぐために、以下のアクションから選択してください。

- --disable-key-availability-check パラメータを使用して、設定ツールの構成ファイルで、キーの可用性を「false」に設定します。詳細については、設定ツールの「AWS CloudHSM クライアント SDK 5 設定パラメータ」セクションを参照してください。
- HSM が 2 つあるクラスターを使用する場合は、コードの初期化中を除いて、エラーの原因となった操作は使用しないでください。
- クラスター内の HSM の数を 3 つ以上に増やしてください。

AWS CloudHSM JCE を使用したキーの抽出

以下のセクションでは、JCE を使用した AWS CloudHSM キーの抽出に関する問題のトラブルシューティングを行います。

GetEncoded、GetPrivateExponent、または getS が null を返します

`getEncoded`、`getPrivateExponent`、`getS` はデフォルトでは無効になっているため、null を返します。これらを有効にするには、「の JCE を使用したキー抽出 AWS CloudHSM」を参照してください。

有効にした後に getEncoded、getPrivateExponent、getS が null を返した場合は、キーが適切な前提条件を満たしていません。詳細については、の JCE を使用したキー抽出 AWS CloudHSM を参照してください。

getEncoded、GetPrivateExponent、または getS は HSM の外部でキーバイトを返します

お客様またはシステムへのアクセス権を持つユーザーがクリアキー抽出を有効にしました。この設定をデフォルトの無効状態にリセットする方法など、詳細については以下のページを参照してください。

- の JCE を使用したキー抽出 AWS CloudHSM
- キーの保護と HSM からの抽出

HSM スロットリング

ワークロードが AWS CloudHSM クラスターのハードウェアセキュリティモジュール (HSM) 容量を超えるHSMs がビジー状態またはスロットリング中であることを示すエラーメッセージが表示されます。この場合、スループットが低下したり、HSM からのリクエストを拒否する割合が高くなることがあります。さらに、HSM は次のビジーエラーを送信する可能性があります。

クライアント SDK 5 向け

- PKCS11 では、ビジーエラーは CKR_FUNCTION_FAILED にマップされます。このエラーは複数の理由で発生する可能性がありますが、HSM スロットリングによってこのエラーが発生すると、ログに次のログ行が表示されます。
  - `[cloudhsm_provider::hsm1::hsm_connection::e2e_encryption::error] Failed to prepare E2E response. Error: Received error response code from Server. Response Code: 187`
  - `[cloudhsm_pkcs11::decryption::aes_gcm] Received error from the server. Error: This operation is already in progress. Internal error code: 0x000000BB`
- JCE では、ビジーエラーは `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected error with the Provider: The HSM could not queue the request for processing.` にマップされます。

- 他の SDK のビジーエラーはメッセージ「Received error response code from Server. Response Code: 187」を出力します。

クライアント SDK 3 向け

- PKCS11 では、ビジーエラーは CKR_OPERATION_ACTIVE にマップされます。
- JCE では、ビジーエラーは 0xBB (187) ステータスとして CFM2Exception にマップされます。アプリケーションは CFM2Exception の getStatus() 関数を使用して HSM からどのようなステータスが返されたかを確認できます。
- 他の SDK のビジーエラーはメッセージ「HSM Error: HSM is already busy generating the keys(or random bytes) for another request.」を出力します。

解決策

この問題は、次の 1 つまたは複数のアクションを実行することで解決できます。

- 拒否された HSM オペレーションに対する再試行コマンドをアプリケーションレイヤーに追加します。再試行コマンドを有効にする前に、クラスターがピーク時の負荷に対応できる適切なサイズになっていることを確認してください。

> **Note**
>
> クライアント SDK 5.8.0 以降では、再試行コマンドはデフォルトでオンになっています。各 SDK の再試行コマンド設定の詳細については、「クライアント SDK 5 設定ツールの詳細設定」を参照してください。

- 「AWS CloudHSM クラスターでの HSMsスケーリング」の手順に従って HSM をクラスターに追加してください。

> **Important**
>
> クラスターの負荷テストを行って予測すべきピーク負荷を決定し、高可用性を確保するためにクラスターに HSM を 1 つ追加することを推奨します。

- アプリケーションが送信するリクエストの数を減らします。クライアント SDK 5 の PKCS #11 ライブラリでは、テンプレートに含まれる属性の数に関係なく、各C_GetAttributeValue呼び出しが 1 つのリクエストを送信します。1 つのテンプレートに必要なすべての属性をリクエストし、

その値をアプリケーションにキャッシュします。詳細については、「AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリを使用して属性を取得する」を参照してください。

JCE では、リクエストする各属性には 1 つのリクエストの費用がかかるため、アプリケーションに必要な属性のみをリクエストします。

AWS CloudHSM クラスターへの接続が失われました

AWS CloudHSM クライアントを設定するときに、クラスター内の最初の HSM の IP アドレスを指定しました。この IP アドレスは、 AWS CloudHSM クライアントの設定ファイルに保存されます。クライアントが起動すると、この IP アドレスへの接続を試みます。接続できない場合 (HSM で障害が発生した、HSM を削除した場合など) は、次のような エラーが表示されることがあります。

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

このような エラーを解決するには、クラスター内のアクティブで到達可能な HSM の IP アドレスを指定して、設定ファイルを更新します。

AWS CloudHSM クライアントの設定ファイルを更新するには

1. 次のいずれかの方法を使用して、クラスター内のアクティブな HSM の IP アドレスを見つけます。

   - 「 コンソール」のクラスター詳細ページに [AWS CloudHSM HSM] を表示します。
   - AWS Command Line Interface (AWS CLI) を使用して describe-clusters コマンドを発行します。

   この IP アドレスは、後の手順で必要になります。
2. クライアントを停止するには、次のコマンドを使用します。

   Amazon Linux

   ```
   $ sudo stop cloudhsm-client
   ```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

Windows

- Windows クライアント 1.1.2+ の場合:

  ```
  C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
  ```

- Windows クライアント 1.1.1 以前の場合。

  AWS CloudHSM クライアントを起動したコマンドウィンドウで Ctrl+C を使用します。

3. クライアントの設定ファイルを更新するには、前のステップで検出した IP アドレスを指定して、次のコマンドを使用します。

```
$ sudo /opt/cloudhsm/bin/configure -a <IP address>
```

4. クライアントを起動するには、次のコマンドを使用します。

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows クライアント 1.1.2+ の場合:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows クライアント 1.1.1 以前の場合。

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

CloudWatch に AWS CloudHSM 監査ログがない

2018 年 1 月 20 日より前に AWS CloudHSM クラスターを作成した場合は、そのクラスターの監査ログの配信を有効にするために、サービスにリンクされたロールを手動で設定する必要があります。HSM クラスターでサービスリンクされたロールを有効にする方法については、サービスリンクされたロールを理解する、および IAM ユーザーガイド の サービスリンクされたロールを作成する を参照してください。

AES キーラップインの長さが非準拠IVs AWS CloudHSM

このトラブルシューティングトピックは、アプリケーションが回復不可能なラップされたキーを生成するかの判断に役立ちます。この問題の影響を受けている場合、このトピックを活用して問題に対処してください。

トピック

- コードが回復不可能なラップされたキーを生成する可能性の判断
- コードが回復不可能なラップされたキーを生成する場合に実行が必要なアクション

コードが回復不可能なラップされたキーを生成する可能性の判断

以下の すべて の条件に当てはまる場合、影響を受けます。

Condition	確認方法
PKCS #11 ライブラリをアプリケーションで使用します	PKCS #11 ライブラリは、`libpkcs11.so` フォルダ内の `/opt/cloudhsm/lib` ファイルとしてインストールされます。C 言語で書かれたアプリケーションは、通常 PKCS #11 ライブラリを直接使用しますが、Java で書かれたアプリケーションは Java 抽象化レイヤーを介して間接的にライブラリを使用する場合があります。Windows を使用している場合、PKCS #11 ライブラリは現在 Windows では利用できないため、影響を受けません。
アプリケーションでは PKCS #11 ライブラリのバージョン 3.0.0 を特に使用します	AWS CloudHSM チームから E メールを受け取った場合は、PKCS #11 ライブラリのバージョン 3.0.0 を使用している可能性があります。 アプリケーションインスタンスのソフトウェアバージョンを確認するには、次のコマンドを使用します。 `rpm -qa | grep ^cloudhsm`
AES キーラッピングを使用してキーをラップします	AES キーラップとは、AES キーを使用して他のキーをラップすることを意味します。対応するメカニズム名は `CKM_AES_KEY_WRAP` です。これは、関数 `C_WrapKey` とともに使用されます。`CKM_AES_GCM` や など、初期化ベクトル (IVs) を使用する他の AES ベースのラッピングメカニズムは`CKM_CLOUDHSM_AES_GCM` 、この問題の影響を受けません。関数とメカニズムの詳細情報
AES キーラッピングを呼び出すときにカスタム IV を指定し、この IV の長さは 8 より短くなります。	AES キーラップは通常、`CK_MECHANISM` の構造を使用して以下の通り初期化されます。

Condition	確認方法
	`CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH};` これは、次の場合にのみ適用されます。 • IV_POINTER は NULL ではありません • IV_LENGTH が 8 バイト未満です

上記の条件をすべて満たさない場合は、今すぐ読み取りを停止することができます。ラップされたキーは適切にアンラップでき、この問題は影響しません。それ以外の場合はthe section called “コードが回復不可能なラップされたキーを生成する場合に実行が必要なアクション”を参照してください。

コードが回復不可能なラップされたキーを生成する場合に実行が必要なアクション

次の 3 つの手順を実行する必要があります。

1. PKCS #11 ライブラリを新しいバージョンにすぐにアップグレードします
   - Amazon Linux、CentOS 6、RHEL 6 用最新 PKCS #11 ライブラリ
   - Amazon Linux 2、CentOS 7、RHEL 7 用最新 PKCS #11 ライブラリ
   - Ubuntu 16.04 LTS 用の最新の PKCS #11 ライブラリ
2. 標準に準拠した IV を使用するためソフトウェアを更新します

   サンプルコードに従い、 NULL IV のみを指定することを強くお勧めします。これにより、HSM は標準準拠のデフォルト IV を使用します。または、IV を 0xA6A6A6A6A6A6A6A6 の対応する IV 長さを持つ 8 として明示的に指定することもできます。AES キーラッピングに他の IV を使用することはお勧めしません。将来のバージョンの PKCS #11 ライブラリでは AES キーラッピングのカスタム IV を明示的に無効にします。

   IV を適切に指定するサンプルコードは、GitHub の aes_wrapping.c に表示されます。
3. 既存のラップされたキーを特定して復元します

PKCS #11 ライブラリのバージョン 3.0.0 を使用してラップしたキーを特定し、キーを回復する場合、サポート (https://aws.amazon.com/support) へお問い合わせください。

> **Important**
>
> この問題は、PKCS #11 ライブラリのバージョン 3.0.0 でラップされたキーにのみ影響します。PKCS #11 ライブラリの以前のバージョン (2.0.4 および番号の小さいパッケージ) またはそれ以降のバージョン (3.0.1 以上の番号のパッケージ) を使用してキーをラップできます。

AWS CloudHSM クラスター作成の失敗の解決

クラスターを作成すると、ロールがまだ存在しない場合、 は AWSServiceRoleForCloudHSM サービスにリンクされたロール AWS CloudHSM を作成します。がサービスにリンクされたロールを作成 AWS CloudHSM できない場合、クラスターの作成が失敗する可能性があります。

このトピックでは、クラスターを正常に作成できるように、代表的な問題の解決方法を示します。このロールは 1 回だけ作成する必要があります。サービスにリンクされたロールをアカウントに作成すると、サポートされている任意の方法を使用して、追加のクラスターを作成および管理できます。

以下のセクションでは、サービスにリンクされたロールに関連するクラスター作成エラーのトラブルシューティング対策を示します。これらの対策を試してもクラスターを作成できない場合は、サポート までお問い合わせください。AWSServiceRoleForCloudHSM サービスにリンクされたロールの詳細については、「のサービスにリンクされたロール AWS CloudHSM」を参照してください。

トピック

- 不足しているアクセス権限の追加
- サービスにリンクされたロールを手動で作成する
- 非フェデレーションユーザーの使用

不足しているアクセス権限の追加

サービスにリンクされたロールを作成するには、`iam:CreateServiceLinkedRole` アクセス権限が必要です。クラスターを作成する IAM ユーザーにこのアクセス許可がない場合、 AWS アカウン

トでサービスにリンクされたロールを作成しようとすると、クラスターの作成プロセスは失敗します。

アクセス権限が不足しているためにエラーが発生すると、以下のエラーメッセージが表示されます。

```
This operation requires that the caller have permission to call
 iam:CreateServiceLinkedRole to create the CloudHSM Service Linked Role.
```

このエラーを解決するには、クラスターを作成する IAM ユーザーに AdministratorAccess アクセス権限を付与するか、ユーザーの IAM ポリシーに iam:CreateServiceLinkedRole アクセス権限を追加します。手順については、「新しいユーザーまたは既存のユーザーへのアクセス権限の追加」を参照してください。

その後で、もう一度クラスターを作成してみてください。

サービスにリンクされたロールを手動で作成する

IAM コンソール、CLI、または API を使用して、AWSServiceRoleForCloudHSM サービスにリンクされたロールを作成できます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの作成」を参照してください。

非フェデレーションユーザーの使用

認証情報の発信元となるフェデレーティッドユーザーは AWS、フェデレーティッド以外のユーザーのタスクの多くを実行できます。ただし、AWS は、サービスにリンクされたロールをフェデレーションエンドポイントから作成するための API コールをユーザーに許可していません。

この問題を解決するには、非フェデレーションユーザーを作成して iam:CreateServiceLinkedRole アクセス権限を付与するか、既存の非フェデレーションユーザーに iam:CreateServiceLinkedRole アクセス権限を付与します。次に、そのユーザーに から クラスターを作成 AWS CLIしてもらいます。これにより、サービスにリンクされたロールがアカウントに作成されます。

サービスにリンクされたロールが作成されたら、必要に応じて、非フェデレーションユーザーが作成したクラスターを削除できます。クラスターを削除しても、ロールには影響しません。その後、フェデレーティッドユーザーを含む必要なアクセス許可を持つユーザーは、アカウントに AWS CloudHSM クラスターを作成できます。

ロールが作成されたことを検証するには、https://console.aws.amazon.com/iam/ で IAM コンソールを開き、ロールを選択します。または、 AWS CLIの IAM get-role コマンドを使用します。

```
$  aws iam get-role --role-name AWSServiceRoleForCloudHSM
 {
     "Role": {
         "Description": "Role for CloudHSM service operations",
         role policy statement
         "RoleId": "AROAJ4I6WN5QVGG5G7CBY",
         "CreateDate": "2017-12-19T20:53:12Z",
         "RoleName": "AWSServiceRoleForCloudHSM",
         "Path": "/aws-service-role/cloudhsm.amazonaws.com/",
         "Arn": "arn:aws:iam::111122223333:role/aws-service-role/
cloudhsm.amazonaws.com/AWSServiceRoleForCloudHSM"
     }
 }
```

AWS CloudHSM クライアント設定ログの取得

AWS CloudHSM では、 クライアント SDK 3 と クライアント SDK 5 が AWS 、サポートが問題のトラブルシューティングを行うために環境に関する情報を収集するためのツールを提供しています。

トピック

- AWS CloudHSM クライアント SDK 5 サポートツール
- AWS CloudHSM クライアント SDK 3 サポートツール

AWS CloudHSM クライアント SDK 5 サポートツール

AWS CloudHSM クライアント SDK 5 のスクリプトは、次の情報を抽出します。

- クライアント SDK 5 コンポーネントの設定ファイル
- 使用可能なログファイル
- オペレーティングシステムの現行バージョン
- パッケージの情報:

クライアント SDK 5 の情報ツールの実行

クライアント SDK 5 には、各コンポーネントのクライアントサポートツールが含まれていますが、すべてのツールは同じ機能を果たします。収集されたすべての情報を含む出力ファイルを作成するツールを実行します。

ツールは次のような構文を使用します。

```
[ pkcs11 | dyn | jce | ksp | cli]_info
```

たとえば、PKCS #11 ライブラリを実行する Linux ホストから対応情報を収集し、システムがデフォルトディレクトリに書き込むようにする場合、次のコマンドを実行します。

```
/opt/cloudhsm/bin/pkcs11_info
```

ツールは /tmp ディレクトリ内に出力ファイルを作成します。

AWS CloudHSM CLI

Linux で CLI AWS CloudHSM のサポートデータを収集するには

- 対応ツールを使用してデータを収集します。

  ```
  /opt/cloudhsm/bin/cli_info
  ```

Windows AWS CloudHSM で のサポートデータを収集するには

- 対応ツールを使用してデータを収集します。

  ```
  PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cli_info.exe"
  ```

PKCS #11 library

Linux で PKCS #11 ライブラリの対応データの収集

- 対応ツールを使用してデータを収集します。

  ```
  /opt/cloudhsm/bin/pkcs11_info
  ```

Windows で PKCS #11 ライブラリの対応データの収集

- 対応ツールを使用してデータを収集します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe"
```

OpenSSL Dynamic Engine

Linux で OpenSSL Dynamic Engine の対応データの収集

- 対応ツールを使用してデータを収集します。

```
/opt/cloudhsm/bin/dyn_info
```

JCE provider

Linux で JCE プロバイダーの対応データの収集

- 対応ツールを使用してデータを収集します。

```
/opt/cloudhsm/bin/jce_info
```

Windows で JCE プロバイダの対応データの収集

- 対応ツールを使用してデータを収集します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe"
```

Key Storage Provider

Windows でキーストレージプロバイダーの対応データの収集

- 対応ツールを使用してデータを収集します。

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\ksp_info.exe"
```

サーバーレス環境からのログの取得

Fargate や Lambda などのサーバーレス環境に を設定するには、 AWS CloudHSM ログタイプを に設定することをお勧めしますterm。term に設定すると、サーバーレス環境は CloudWatch に出力できるようになります。

CloudWatch からクライアントログを収集するには、Amazon CloudWatch Logs ユーザーガイドの「ロググループとログストリームの操作」を参照してください。

AWS CloudHSM クライアント SDK 3 サポートツール

AWS CloudHSM クライアント SDK 3 のスクリプトは、次の情報を抽出します。

- オペレーティングシステムとその現在のバージョン
- cloudhsm_client.cfg、cloudhsm_mgmt_util.cfg、application.cfg ファイルからのクライアント設定情報
- プラットフォームに固有の場所からのクライアントログ
- cloudhsm_mgmt_util を使用してのクラスターと HSM 情報
- OpenSSL の情報
- 現在のクライアントとビルドのバージョン
- インストーラのバージョン

クライアント SDK 3 の情報ツールの実行

このスクリプトは、収集されたすべての情報を含む出力ファイルを作成します。スクリプトは /tmp ディレクトリ内に出力ファイルを作成します。

Linux: /opt/cloudhsm/bin/client_info

Windows: C:\Program Files\Amazon\CloudHSM\client_info

> **Warning**
>
> このスクリプトには、クライアント SDK 3 バージョン 3.1.0 ~ 3.3.1 に関する既知の問題があります。この問題の修正を含むバージョン 3.3.2 にアップグレードすることを強くお勧めします。このツールを使用する前の詳細情報は、既知の問題 ページを参照してください。

AWS CloudHSM クォータ

クォータは、以前は制限と呼ばれていましたが、 AWS リソースに割り当てられた値です。 AWS リージョンと AWS アカウントごとの AWS CloudHSM リソースには、次のクォータが適用されます。デフォルトのクォータは によって適用される初期値であり AWS、これらの値を以下の表に示します。調整可能なクォータは、デフォルトのクォータよりも大きくすることができます。

Service Quotas

リソース	デフォルトのクォータ	引き上げ可能?
クラスター	4	はい
HSM	6	はい
クラスターあたりの HSM	28	いいえ

クォータの増加をリクエストする方法として推奨されるのは、Service Quotas コンソールを開く方法です。このコンソールで、サービスとクォータを選択し、リクエストを送信します。詳細については、Service Quotas ドキュメントを参照してください。

次のシステムクォータ表のクォータは調整できません。

システムクォータ

[リソース]	hsm1.medium のクォータ	hsm2m.medium のクォータ
クラスターあたりの最大キー	3,300	キー総数 16,666 個、非対称キーの最大数 3,333 個
クラスターあたりの最大ユーザー数	250	1,024
ユーザー名の最大長	31 文字	31 文字
必要なパスワードの長さ	8 ~ 32 文字	8 ~ 32 文字
クラスターあたりの同時クライアント接続の最大数[1]	900	900

[リソース]	hsm1.medium のクォータ	hsm2m.medium のクォータ
アプリケーションあたりの PKCS #11 セッションの最大数	1,024	1,024
クラスターあたりの最大保存証明書数[2]	サポートされていません	60
HSM あたりの 1 秒あたりの証明書ストレージ読み取りオペレーション[2]	サポートされていません	10
HSM あたりの 1 秒あたりの証明書ストレージ書き込みオペレーション[2]	サポートされていません	1

[1] クライアント SDK 3 のクライアント接続はクライアントデーモンです。クライアント SDK 5 では、クライアント接続はアプリケーションです。

[2] 証明書ストレージは hsm2m.medium クラスターでのみ使用できます。詳細については、「証明書ストレージの制限」を参照してください。

AWS CloudHSM クライアント SDK のダウンロード

以下のトピックでは、 AWS CloudHSM クライアント SDKs のダウンロードについて説明します。

> **Note**
>
> 各 クライアント SDK がサポートするプラットフォームについては、「AWS CloudHSM クライアント SDK 5 がサポートするプラットフォーム」と「AWS CloudHSM クライアント SDK 3 でサポートされているプラットフォーム」を参照してください。

トピック

- AWS CloudHSM 最新の Client SDK リリース
- AWS CloudHSM 以前の Client SDK リリース
- AWS CloudHSM 非推奨の Client SDK リリース
- クライアント SDK AWS CloudHSM end-of-lifeリリース

AWS CloudHSM 最新の Client SDK リリース

2021 年 3 月、 クライアント SDK バージョン 5.0.0 が AWS CloudHSM リリースされました。これにより、さまざまな要件、機能、プラットフォームサポートを備えたまったく新しいクライアント SDK が導入されました。

クライアント SDK 5 は本番環境で完全にサポートされており、クライアント SDK 3 と同じコンポーネントとレベルのサポートを提供します。詳細については、「AWS CloudHSM クライアント SDK コンポーネントのサポートを比較する」を参照してください。

> **Note**
>
> CloudHSM SDK 5.17 以降、SDK バージョンのリリース日から最大 3 つの以前のマイナーバージョンと 1 年をサポートするようにサービスレベルアグリーメントを更新しています。CloudHSM SDK の新しいバージョンをリリースして、新機能を提供し、セキュリティ、安定性、パフォーマンスの修正を行います。新しいバージョンが利用可能になると、古いバージョンとサポートされていないバージョンの CloudHSM SDK のダウンロードリンクは無効になります。

このセクションでは、クライアント SDK の最新バージョンについて説明します。以前のリリースの詳細については、「以前の SDK リリース」を参照してください。廃止されたバージョンについては、「廃止された SDK リリース」を参照してください。

クライアント SDK 5 リリース: バージョン 5.18.0

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 8d321bd3d74b7365e40107e43e5aa4680f4938513ca1c33f1c739369d54cc466)
- OpenSSL Dynamic Engine (SHA256 checksum 107d522466e83e6f9a9d92d189271180ed60b2bf0d7343e0d46225106a485480)
- OpenSSL プロバイダー (SHA256 checksum 0d6470a0083273ae1d0922ab7b8b240f929762764bd7b0c84d95f87b3c38fca4)
- JCE プロバイダー(SHA256 checksum 2ee2076bea49b970e1b9f6a0edee11dc9cbd139ef11a2687e4e6f54c9cc3ef66)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 0c77a4014d77df64ea3faaf1cc8d8f67dacd8a5ca53baa0026ab0b65d4fe6b69)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 8206a783676c2dc737872f8d536d2233e76197505b80c5bbff11690cc975d105)
- OpenSSL Dynamic Engine (SHA256 checksum 5eb84a69a9f28218660ec6d879b86964da3402dc27ecaacf205316723346023d)
- OpenSSL プロバイダー (SHA256 checksum a6cafef403e45c19faf6977906ffbf06b4d12b79ddb5fb1f26383c9d1e3dd4ec)
- JCE プロバイダー(SHA256 checksum 8323afdfd2f080dad51c92fe6fd1b802c7db76f6b29949d6c5349472e1377a06)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum a709088dd8857988c893475d85b9a1b3d7a3545006b119156d1e45bebe77021e)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 3d6de0aac5e9f49e96f8aace1e2515c050934f76952ae2a74b3a15952fe9b01e)
- OpenSSL Dynamic Engine (SHA256 checksum c68bbffc1b26926072f0b4e066c03e6f4721d36f4f2ea184efae053dd5088f30)
- JCE プロバイダー(SHA256 checksum a1157f1a66a43c90ba3bd79a1dabbb2435c66f8ae50d096644aecf40a4392205)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum f29d3ad3fb8084c2a6491003907f494876af800b3a925a000ef11702b05be306)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 5fb27847ed54c4ec19dbe395eab417b5fe79071f3560a5e0200240bb540542e7)
- OpenSSL Dynamic Engine (SHA256 checksum e93c7ac45d754ba9217ed68cd9f2c09c3581128518119a3edada75c36fedd18a)
- JCE プロバイダー(SHA256 checksum 19741d2bd33e9b540379878baf2917b0fe95fa14a103e3192dad0050028bb032)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 8e9427916230ee4b6357620c310a5463d001342a2955c436e93e314a369fbc26)

RHEL 10 (10.0+)

x86_64 アーキテクチャの RHEL 10 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 56c835581b68fbf6fff052f43fad93c60986c82cc74ee012dcaba25a4153ca06)
- OpenSSL プロバイダー (SHA256 checksum a06765c7e10ac46125a27ae4466464efb28d722decadb90881d6b002a7949954)
- JCE プロバイダー(SHA256 checksum 86b2e64b8d5b354b2ea0e41126f2cccea1be124160621d04e7ee534d2570172b)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs

- CloudHSM CLI (SHA256 checksum 5eabb49d6be32f2dc6e9c9a377bd0d6fc2354eb3b51412e0002fa061625ff206)

ARM64 アーキテクチャの RHEL 10 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 96b3ae6cf52b63369fe1420ae4ed95a2bf2f42b7c1543d87894ff6014bece1ea)
- OpenSSL プロバイダー (SHA256 checksum dc6b4e852f8a079ad98fc6af11a177cd8e738ab89bd6ccee0392c25898cf3ef2)
- JCE プロバイダー(SHA256 checksum 6f93577ca3dd2f5ae0980be8811753f294df193d36583f42a92d859c9a6143e2)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum a4dd39dbdf4ffd6e589b485635e535a3d0fbf4cab815378ff7d4fa1576c3de17)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum ef2d3a0d427f6d246a69db06a07b520d98721f24fa1180af2a70a6cb90f175b8)
- OpenSSL Dynamic Engine (SHA256 checksum 56e67217c93bcaca96aafede8049a83018740edcaedb72cc7801350be715b28c)
- OpenSSL プロバイダー (SHA256 checksum b4fe26cc8159f77188430238660a64c6cb8ff6db21c7945ce62ba2832d3c3610)
- JCE プロバイダー(SHA256 checksum 481e2cda3c26b5297f91402e80176350feccbb9f1b58001f1f577d1c800340dd)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d37cabf2373a9935394339f704b6f92bf183fbe8abcb2ab92deb756e0fec71ee)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b0d8ec4871887ad40c51e725d510137ed7311eecfc3981060eea74d99fda4ab5)
- OpenSSL Dynamic Engine (SHA256 checksum 475404fd30d008f952e2d07c8fc39aaeb6de200248e3cfa07711a3dbcb2d7d04)

- OpenSSL プロバイダー (SHA256 checksum 2985c6f422a2dc21578cd6f15876c147543776ba7da91db5990621543efc5ff8)
- JCE プロバイダー(SHA256 checksum 45a035f5005459431a68786ea4db0184b71f8e8d5277db8b8ccbe251416b10ca)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 803a867e3df2f2aeb5230f6939db8ffd202991406bce2c626b987465f802fcfb)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 912d5d2a7376180124283309 3472d1365cbcfc3551b1fe93743dce0cbdf8c727)
- OpenSSL Dynamic Engine (SHA256 checksum 70b9aedde526fed0d65114f9b4d8a9576b1ad0efe0bc271c7ef6765345ef66ce)
- JCE プロバイダー(SHA256 checksum ecc9146e3ba72af288834bffe5c4a93922e9ae379f9663fac7d86ffbb36785a8)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 784009c529ff86f357b20c62f1680f35a4710e22d1fd40f9018dab81fa0030c7)

ARM64 アーキテクチャの RHEL 8 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 9ee50fd41904ade515bfab6073fa2d8ba2e1e1c373116420d9125049e3bcdd1f)
- OpenSSL Dynamic Engine (SHA256 checksum 26ac4952f6363ba5a5fcebd45b9709343ed31d05fc5c21240b0e530520295caa)
- JCE プロバイダー(SHA256 checksum ec2c5e4aa8ee0307cf393ad14a890dab5ed904aa4e0de7f7d9ec0806c243a704)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum edbeec5c1f8ac877fbd53f70dbe906bf69839a0493802bf9b1ef1c3c574a07e3)

Ubuntu 26.04 LTS

x86_64 アーキテクチャの Ubuntu 26.04 LTS 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 4b3864459b482f8d3862582b52809315b7653b8b63523f47920c3812643cbab0)
- OpenSSL Dynamic Engine (SHA256 checksum 3e53d5da4ab44384b90bcdc5a4b9b120a630f407df633765414dc3146d9e6e28)
- OpenSSL プロバイダー (SHA256 checksum 4967595c4ea3538f13c647469f9a7ba4583b54bb81ca761bbe1488bcfca78395)
- JCE プロバイダー(SHA256 checksum bc24b988be625826a18353dd0d2c2b1366e28d6d57386865bb8dc6b1c38ed794)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 70b18fad5e3f150695a6e476c13b3e5efa29a5c0c38010522737e0ad36f4cba0)

ARM64 アーキテクチャの Ubuntu 26.04 LTS 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 23d562db2c6fc6a9253a66c32eb11283f7300aada9c4a714516f1dce029a731a)
- OpenSSL Dynamic Engine (SHA256 checksum dfc2777e34d6016c9f39edf97553cd23591b34856bfae56cfead25a4969e9eb7)
- OpenSSL プロバイダー (SHA256 checksum 9f2e8ffbd92334942aea1df515c0f987c88c698acd4a74e03f7a14fe98c682be)
- JCE プロバイダー(SHA256 checksum c85a67409cbbbd38f54b6ac2f2ab8a67335452d42b03628681750b08bf938ebe)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6fa5d24febe3e3f6c8649f3bd75e2b090660cd82fbedac0e5491a726988f99bc)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 72ab424e34ee2780ae770679180f456374d8be1b66a82aa441defa603c8ec3a7)
- OpenSSL Dynamic Engine (SHA256 checksum e9ebd0cada051ff21218655f1431171ddd69d219d3cdea13fd9b72ac30fb0a0c)
- JCE プロバイダー(SHA256 checksum 551277453f0bf3a8e78a0e0fec31eab3341dd40adceff3225e606800e0c9c12e)

    - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 28cba68e9721a1683f9a394704bd786b234cb44a8acc10dbe696aaee351a6f8d)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum c17653c12df1448cba9f1714c6325e35d51df4015557db26b239b82cc9d50db1)
- OpenSSL Dynamic Engine (SHA256 checksum 253c3b8b3e40461889e5101d66f88fa7d21866ee9af1155e014890f6bb795ad1)
- JCE プロバイダー(SHA256 checksum 1e79f34dbc840ec605cb85b4dcd61a01aa2490200e13c98f5021191f4a3bbb41)
    - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 150a3ce22f9a2e7e6ac1b2415b94edc09c7589a64044856409071f0a55360b2d)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7c1f6c82edfa7ecbd98ee8e27d4fca205aacdd770ebfe962511214d511beb904)
- OpenSSL Dynamic Engine (SHA256 checksum 014106115154a1d47a44be7dbbeb0483b8c2ed7e676e728c1bb0d73323a94d00)
- JCE プロバイダー(SHA256 checksum 0ff395b8ff58832b2b50f82066e8979329cb95ae4097748632a9b590e88e352a)
    - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 8e5cc97c69bea415a272eae5b85d16566342cb581441a5061042a3998dc5600f)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7c5e51bb931f54cd524d5c02487b3c2e0c26d802cf52c1d6dee8f6f1dde5513b)

- OpenSSL Dynamic Engine (SHA256 checksum 7a3a1b696fe1fcb34b3a3c4f4556fa89038d27165a1265458289f9b67694422a)
- JCE プロバイダー(SHA256 checksum ce9740e58424e6e2f2cc98d840755e440339465da323e85049ab7dd2edc4be46)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 20ec9d5dd380be3f77fec871b45e1e7739b00b9438b95213666d0e3173a99d76)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7a6486f73ad63312ef4b09b5284185f666d68abf5c5cd23c0a234a071e1d59e6)
- JCE プロバイダー(SHA256 checksum 89dc221ffce8510b7b1365d5655e4b6823341986d4ecb159350e9e1e8a02e823)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b536e61880080695dc705234eb45abef5f34bc20273ae9cdca25acf9163d86a6)
- キーストレージプロバイダー (KSP) (SHA256 checksum 8dd827b51e471e08094c963b745cd6960d4d398b0ee26930981bafccaf4258a6)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7a6486f73ad63312ef4b09b5284185f666d68abf5c5cd23c0a234a071e1d59e6)
- JCE プロバイダー(SHA256 checksum 89dc221ffce8510b7b1365d5655e4b6823341986d4ecb159350e9e1e8a02e823)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b536e61880080695dc705234eb45abef5f34bc20273ae9cdca25acf9163d86a6)
- キーストレージプロバイダー (KSP) (SHA256 checksum 8dd827b51e471e08094c963b745cd6960d4d398b0ee26930981bafccaf4258a6)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7a6486f73ad63312ef4b09b5284185f666d68abf5c5cd23c0a234a071e1d59e6)
- JCE プロバイダー(SHA256 checksum 89dc221ffce8510b7b1365d5655e4b6823341986d4ecb159350e9e1e8a02e823)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b536e61880080695dc705234eb45abef5f34bc20273ae9cdca25acf9163d86a6)
- キーストレージプロバイダー (KSP) (SHA256 checksum 8dd827b51e471e08094c963b745cd6960d4d398b0ee26930981bafccaf4258a6)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.18.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7a6486f73ad63312ef4b09b5284185f666d68abf5c5cd23c0a234a071e1d59e6)
- JCE プロバイダー(SHA256 checksum 89dc221ffce8510b7b1365d5655e4b6823341986d4ecb159350e9e1e8a02e823)
  - AWS CloudHSM (SHA256 checksum 741a56da7358118b091ed8cdcecbccc445574688249b1d5cfba4ae5d153d16c2) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b536e61880080695dc705234eb45abef5f34bc20273ae9cdca25acf9163d86a6)
- キーストレージプロバイダー (KSP) (SHA256 checksum 8dd827b51e471e08094c963b745cd6960d4d398b0ee26930981bafccaf4258a6)

クライアント SDK 5.18.0 では、ポスト量子 ML-DSA 署名や EdDSA サポートなど、複数のコンポーネントに新機能と改善が導入されています。

プラットフォームのサポート

- Ubuntu 26.04 LTS サポートを追加しました。

新機能

- 量子キーの生成、署名、検証後の ML-DSA (Module-Lattice-Based Digital Signature Algorithm、FIPS 204) のサポートが追加されました。
- EdDSA (Edwards-curve デジタル署名アルゴリズム) の署名と検証のサポートが追加されました。

CloudHSM CLI

- ML-DSA キーペア (ML-DSA-44、ML-DSA-65、ML-DSA-87) の生成、署名、検証のサポートが追加されました。詳細については、CloudHSM CLI を使用して非対称 ML-DSA キーペアを生成する、CloudHSM CLI で ML-DSA メカニズムを使用して署名を生成する、および CloudHSM CLI で ML-DSA メカニズムで署名された署名を検証する を参照してください。
- PureEdDSA (Ed25519) の署名と検証を追加し、CloudHSM CLI での EdDSA サポートを拡張しました。詳細については、「CloudHSM CLI で PureEdDSA メカニズムを使用して署名を生成する」および「CloudHSM CLI で PureEdDSA メカニズムで署名された署名を検証する」を参照してください。

PKCS #11 ライブラリ

- ML-DSA キーの生成、署名、検証、および EdDSA (Ed25519 および Ed25519ph) の署名と検証のサポートが追加されました。詳細については、「AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされているメカニズム」を参照してください。

JCE プロバイダー

- ML-DSA キーの生成、署名、検証、および EdDSA (Ed25519 および Ed25519ph) のサポートが追加されました。詳細については、「AWS CloudHSM クライアント SDK 5 の JCE プロバイダーでサポートされているメカニズム」を参照してください。

OpenSSL プロバイダー

- 非 FIPS クラスターでの TLS オフロード用の PureEdDSA (Ed25519) キータイプと、FIPS クラスターと非 FIPS クラスターの両方で TLS オフロード用の ML-DSA (ML-DSA-44、ML-DSA-65、ML-DSA-87) キータイプのサポートが追加されました。詳細については、「AWS CloudHSM クライアント SDK 5 の OpenSSL プロバイダーでサポートされているキータイプ」および「OpenSSL プロバイダーがサポートするメカニズム」を参照してください。

- OpenSSL プロバイダーは、サポートされているすべてのキータイプに対して、証明書署名リクエスト (CSR) の作成や証明書署名を含む OpenSSL CLI オペレーションをサポートするようになりました。

バグ修正/機能向上

- CloudHSM CLI および JCE プロバイダーで`key-reference`フィルターがセッション (エフェメラル) キーを選択できなかったという既知の問題を解決しました。
- クライアント SDKs、追加のバグ修正と安定性が向上しました。

AWS CloudHSM 以前の Client SDK リリース

このセクションでは、以前の クライアント SDK のリリースを一覧表示します。

バージョン 5.17.2

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 61bed6310a97119ef1bd7da177fc8f3fdf3ea688468148c08e186428a500b6b2)
- OpenSSL Dynamic Engine (SHA256 checksum 22e7636ca6ef41cbfaad6c226b58a162ba611a2ed950d29a27fde60f7836cb73)
- OpenSSL プロバイダー (SHA256 checksum 1940e89e71f9099dfc6febf27e9afed9824c9255959c752d6d2c953b35229298)
- JCE プロバイダー(SHA256 checksum 78d744b4f10698603f5ea483641b05a35076c4b7627dee245eed777fa5a205e2)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b4bce34c0b74d5d8ae2b59c377246cd176f01cd13a3f771bd048a6eeeed2d695)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 01dda2dbda06216d63447e3119e87e2c729e50326910736fa97d95134a7ccaac)
- OpenSSL Dynamic Engine (SHA256 checksum c0eedc2803f16e0cbbc5c024763b24940860afd70e4c561a40769ab7597847ef)
- OpenSSL プロバイダー (SHA256 checksum b46e600f1c8210283fe77fe66ebc50483590d65c54f38d0d4f8c6caa72a1f02a)
- JCE プロバイダー(SHA256 checksum f3309a72f045e63a61a05b1a1c133099438a06dda6e57e1cb013e90cb383bfaa)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b35aa1c023d6ea3d1b4699aa6e1a96da998f10b9d2aa201bd02b3e7b906cd6b4)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b5bcb6977373d3affa6dab18d05921e665ff8a4c72ff9396711d2817e27080a7)
- OpenSSL Dynamic Engine (SHA256 checksum 603c8c034b4e154c6e022d1a9455d9f0d5dd86f9a25479c7499ed6f680668199)
- JCE プロバイダー(SHA256 checksum 9bcdc6580e4c3c3cf5b762a8b3a5a722693b1b784b487ebe34e625b6615d47d4)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 33af8335abe8eeb00540123cb5a120d3533099b69812038ba02da2e316327c83)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 1839a2c51dcd7f4b81ffd7e493d2aecfd70276d0392cef32aa421e1a8e9d7819)
- OpenSSL Dynamic Engine (SHA256 checksum bcf34ead09ba23debf2aea7859b4afd650732f8dbbfdddec675b68690ea0d775)
- JCE プロバイダー(SHA256 checksum 5afb78d35db24d7c954aac98b69504b7857d834d3c8db0d27a5bcab834cbe44f)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs

- CloudHSM CLI (SHA256 checksum b2940f697d8d4341f9f0b9d8619d9e7663d3a3cb24c6fd4aedc167e1c5acf22d)

RHEL 10 (10.0+)

x86_64 アーキテクチャの RHEL 10 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 0a679176689208e74d90db0b9d3096389617ec6fe7dafefcaa7ca96ad885c3d1)
- OpenSSL プロバイダー (SHA256 checksum c497ff802f377ef15d5f6c320e59f122852521847268c9cac667beb0d52bb1d0)
- JCE プロバイダー(SHA256 checksum 5333438d932b16240291c8d0d96dde19d76487c0c741c2e52764b5a2386b0ce4)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 7fd322439bd23b5eddd2881cd6a30ab43e81fef3659fad8245eaba64f8cf83da)

ARM64 アーキテクチャの RHEL 10 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum d724b1093fb6bf28141ef380708fe0422597af8c285dbeac5118f78d58d92d0a)
- OpenSSL プロバイダー (SHA256 checksum 0f95006198bc6b1fbf98b3fd83f73cf03db98387f3fc0fcf5bc3b92afdf3ecff)
- JCE プロバイダー(SHA256 checksum 9ff17247153659b6a3e0e4d14aebdcc3c74ccf6de213e914525b2ae83b6b8279)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 7b6e9351b51e6cdbb4711d55e15a8f8a617ddaf68375df580dc5806b662e95fd)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 1e65fdbd6a413a160fb0d0927b80ff3b106c287ee43485bda7a7344997319cf0)
- OpenSSL Dynamic Engine (SHA256 checksum 059d5e03badda05e57792f863edb931cfd5b25ac7a8c2087760173b90e476284)

- OpenSSL プロバイダー (SHA256 checksum e3f0e2ef1c3330597b90567e339f99a88512cf9b6c750de64b58efe6a58ee3cb)
- JCE プロバイダー(SHA256 checksum 76cdb538108a56b3c03bc451f66364928c5823e9dd7d8e66f08429dd275fca16)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 748132c6a26567b0ce2889e72f327bc7c1c462361f9db992ed2e61490b5339d8)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 79f5c59fe85f19f16d1863994f19a52fa42f572ff8cb2484421cbf8e231be43c)
- OpenSSL Dynamic Engine (SHA256 checksum d3eaf1fb4d26cb60853930e437d1398d538ce9f4a16b4df9f9af5580f8993123)
- OpenSSL プロバイダー (SHA256 checksum 21f1a0231784e64866b9bf77d2ffa442d1d0d76d86042ba43d38b7154a029b62)
- JCE プロバイダー(SHA256 checksum d4691de51f3ba88563e30bbff8e785cc440d397839891e95655eb7c545dea492)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 80c7511a82dab0bd8f88325bed55e2eedd346f69ab56d4d9b3f68d3ff9b99897)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b3c8a8edfdf7896b011cf0ea323131bd2c1cccaba973abe1b66e5086f9ed571c)
- OpenSSL Dynamic Engine (SHA256 checksum da00cda535bb2959f5927f9671ee33d8c33bbb0e3b1cb092bdf9c001c061d529)
- JCE プロバイダー(SHA256 checksum ede280e95c5635e3410c5a953b751e76af4d24dc392c52ef6126411b30cc6e02)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum dfbf3b26ab1e674943bcb1244f59f93309eb14d2b1f69375385198cd5d946589)

ARM64 アーキテクチャの RHEL 8 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 52b7834d0a383f4dd266eb24606088f2918124b3b131684d249239a8a33ccdef)
- OpenSSL Dynamic Engine (SHA256 checksum 16c145cbb0ecaea05ebb0990c8732f1c75316ca0a77c4d5951fada6e27be17df)
- JCE プロバイダー(SHA256 checksum f2e4e094f9cf7f0b05e3f3607b2613ab713c8893bd8292b116389c2be774b797)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 2594a3c18e0b38dc3239720d4e475dbf7fb91aa360696029477327e2d4242b2c)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 3569462455b3e1ba6d2297f00615d1da62913376016584681f813d4b6c671ce3)
- OpenSSL Dynamic Engine (SHA256 checksum d805bace899b9d595152c4d44286a797b48b1d47b4a6002beac923d249dd8c8f)
- OpenSSL プロバイダー (SHA256 checksum 5a931ef08394c08681913a3d8daed58ee4b1bb55fdf109a8de366cea19fcc561)
- JCE プロバイダー(SHA256 checksum 8343b7b26ddc2dde6daaf382f1d09c46f4b0633ce17112ff3839c717ad8d6335)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum ef5e6d07c71f8d370207e5ca93425e07d1821bc571535890145422401a43752d)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 2c9ed8f60fe8ea5d98753778f2a92384047e18a113ab0dec03a2f5007c03805a)
- OpenSSL Dynamic Engine (SHA256 checksum fd7eda127d4c9d26d9aa880b4fad93947a9f012aa6f89debed7138f77a0ff6cc)
- OpenSSL プロバイダー (SHA256 checksum 97fecfddbcf348840398fb793a1b84e432a7dd8e300161180b4568ea06c087b8)

- JCE プロバイダー(SHA256 checksum 069111cc8757e63ac80d0e20715751291f27effb47938f185d5c21ca60c01892)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 89833ec8e0fa0bcadc92f2784003fa030d26db8aa338f2e8a384e168dfa31a63)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 05e48c9625ed6ea5b86527058dee6282034a36865017991fbd114df23bfea759)
- OpenSSL Dynamic Engine (SHA256 checksum 63f28d02069b312aeb917b8182c9c6e618ab752a4ea86449411d85bfa5c29d17)
- JCE プロバイダー(SHA256 checksum 7fd8bb608d4d08c1f40d4e2b816fe405d50b931840d8d185988059608bb617bd)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 9873c6f73832ac49010376544620ac29862d71bbf23df8d12dcb642fe805d1df)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 4d48f1185fad4a50611ac28ce35e0e1741c3392700f40f63d483c424ec05cd3f)
- OpenSSL Dynamic Engine (SHA256 checksum 1d72a33854c5a88662490bb3ff6794bf238662d42869523a7eb4b31ddcdb97d5)
- JCE プロバイダー(SHA256 checksum 681875ab46c8d738781df35970c99bbc064a1a7155bfd15d4048e53eea8e3e8f)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 2b9acb97054a55fb64b791b09b738bd1c5d433a8a1724f0677f704f8b8df888d)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e08552078e12792ddda780fb2bb3ac98c9f60b3f1b3fec410d627114cc1dbb24)
- JCE プロバイダー(SHA256 checksum 323fc72e3df00f2ec65d17c04ca51acec4b3097c39624e5012019f9af07b09b5)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum cc6a183e6c00f0713ab64d1fbfd01d4828ffba086f407025b5b7140c42012d1a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 5165f44f10eb8be3e99ce89d11206cec1b006d61df2004a022ddf97afe2c98c3)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e08552078e12792ddda780fb2bb3ac98c9f60b3f1b3fec410d627114cc1dbb24)
- JCE プロバイダー(SHA256 checksum 323fc72e3df00f2ec65d17c04ca51acec4b3097c39624e5012019f9af07b09b5)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum cc6a183e6c00f0713ab64d1fbfd01d4828ffba086f407025b5b7140c42012d1a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 5165f44f10eb8be3e99ce89d11206cec1b006d61df2004a022ddf97afe2c98c3)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e08552078e12792ddda780fb2bb3ac98c9f60b3f1b3fec410d627114cc1dbb24)
- JCE プロバイダー(SHA256 checksum 323fc72e3df00f2ec65d17c04ca51acec4b3097c39624e5012019f9af07b09b5)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum cc6a183e6c00f0713ab64d1fbfd01d4828ffba086f407025b5b7140c42012d1a)

- キーストレージプロバイダー (KSP) (SHA256 checksum 5165f44f10eb8be3e99ce89d11206cec1b006d61df2004a022ddf97afe2c98c3)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.17.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e08552078e12792ddda780fb2bb3ac98c9f60b3f1b3fec410d627114cc1dbb24)
- JCE プロバイダー(SHA256 checksum 323fc72e3df00f2ec65d17c04ca51acec4b3097c39624e5012019f9af07b09b5)
  - AWS CloudHSM (SHA256 checksum ddfa01e827443b56362ec32344d605f67755e332b66184a1b0b3b4242a23f087) 用の Javadocs
- CloudHSM CLI (SHA256 checksum cc6a183e6c00f0713ab64d1fbfd01d4828ffba086f407025b5b7140c42012d1a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 5165f44f10eb8be3e99ce89d11206cec1b006d61df2004a022ddf97afe2c98c3)

クライアント SDK 5.17.2 には、バグ修正と改善が含まれています。

バージョン 5.17.1

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e5d37660312c8d5b9c36a9b74555222a755276e2c1f360418c34125b2b715b97)
- OpenSSL Dynamic Engine (SHA256 checksum 9b84d756e56f19b830adbd46aa5ac5096ae28944e34dd8dac3bd534e487d7aee)
- OpenSSL プロバイダー (SHA256 checksum 5df0647ea5fe6da453bbf9a9578de9674ba93b722ec17dfe79825006211dc7a9)
- JCE プロバイダー(SHA256 checksum ac820e95618bf6cecf76b6ae0c5fb59198041f1d583178105a506d255d886fc9)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 3b2f966c05690c14c325ea37cf9799bad477b2573f1077a5319d92933af9e590)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 8cf4f7e9bed86ed15198e0870075fa01cb8243d3646ae9a1417ee3eba0cb3111)
- OpenSSL Dynamic Engine (SHA256 checksum 831c7565caaea18c9c239b060347ec1b0e5121ec94d5bfe4cecda0fd4b1aa9ab)
- OpenSSL プロバイダー (SHA256 checksum 73e91db43c98fa1eac6ce4d39f211d0dafa8445941b9f3adf77876cbeb443561)
- JCE プロバイダー(SHA256 checksum 1cc6bd9226ac98ebf898498f3b51b41b3ed58304f0290bd2045ab472e3bf3e14)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d716108dfc8f0e89e082603dbd5c95379b9f593628ab30bc611c37c169fae8d0)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum c22875d9a81b2a5814d949b06d1467d4fbb111e67a6b29d83e961d95bdebd9e9)
- OpenSSL Dynamic Engine (SHA256 checksum f54d264ede8624867c3010d6b3979c51cd05a34d90a4169208400013c8de76c5)
- JCE プロバイダー(SHA256 checksum c46297760a1d306541498e6a0295d7143273286dbb798c1c65bee38799b68e6b)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 87162212c93d8536d9247632221d855e841637a786c405eb47bc1410d62987d9)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 0abe258732376b20730c2d2e089458e83eff4e751df8e60c0c70d3d92e75d7a4)
- OpenSSL Dynamic Engine (SHA256 checksum 4f8101debd678327a26246e50df10fe79663fc264fefd49deef63e036221fc0b)

- JCE プロバイダー(SHA256 checksum dfc83b8f2dbee97eab98bff431c3374181a0051884abe7e9424621fcdba9c8c5)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum bc43b221610d1c220ed402ee14bb0275ad79fb21a342639d1266579881da953a)

RHEL 10 (10.0+)

x86_64 アーキテクチャの RHEL 10 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 11076479568454ad1083bb6a81953a73f9175f6d8626e247e1e16439a5c105ec)
- OpenSSL プロバイダー (SHA256 checksum 6c3d7a58b38ef39c7b5310958eb684feed20edf6731603a8b727b4bd978f465f)
- JCE プロバイダー(SHA256 checksum 921883948344a06716d6b3b30f67708dedff550978efce6d5817b36c83e83b35)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 4d69481cc4e725e65811e031fe344cb178016cd268759bb910acdb14df9be413)

ARM64 アーキテクチャの RHEL 10 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7a20af9fbe7a9874132c1314841ef2ead0daef92ced3ba2289938a88d0ee929e)
- OpenSSL プロバイダー (SHA256 checksum e762bb21086131725e9ca13b7d25e6fcb474888ae49317e1b045ddf826a8338c)
- JCE プロバイダー(SHA256 checksum c8bb96ded1abdfb54cd710fade3c91ce6030f5dc9ddeea639cc0a0cac5c26bb1)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d48ccaeccc9d4247f0cf4c81a6f092704630933dbc89e0b0ba9a75629fd863b7)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum df4f1de8e68e363b351770dd74782d204f04d4db646c554fa96182b429006618)
- OpenSSL Dynamic Engine (SHA256 checksum 06d2e65adb076e65b81b51bcea34b64f718b6aa29c173eebcaf277de2c8d03ec)
- OpenSSL プロバイダー (SHA256 checksum 802e608484ab748eccdce8da0854dc668e90578d4c2bc6ede4caf4121d10171f)
- JCE プロバイダー(SHA256 checksum 1a4fdfe82cc79a4de85a19d6645916b6b388c919a71a22efae7bafaba195db0f)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum daab5b758c61f49b34b6f7e288debb121adc3d63b9652db086855d160c988570)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 2fe6549339b743200c6be91b8abf8017c3f67c90f7db25ffd0ab1915b91d0f78)
- OpenSSL Dynamic Engine (SHA256 checksum a0a964dcb817251faafcdd432e6260dff1d6e1bcdd7e7ffbd54a691cbe366b58)
- OpenSSL プロバイダー (SHA256 checksum bbcc9b3a2455328e6e7b11ff64b7e7653757960a74bfd9d6b3cc998b2a955fb2)
- JCE プロバイダー(SHA256 checksum f9c74e2ededb9d2fdb7076789223c4cde7a0845b6f999b006fa741e9428195c7)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 0c4c1c7f4e06ba3b5b5dfe619abb0fa9d4729c87c668361fdb99b2a62f4b00fc)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 0f8465ab44f7388980b22bc1f8c6c3c3e09e809ba2712170c85db86918817803)
- OpenSSL Dynamic Engine (SHA256 checksum 8fe4f9fb6088d2a32e210696775d395b918ee867dede8b2152ff37d0d99cc8f3)
- JCE プロバイダー(SHA256 checksum 948d37f435e7b03ec8409ee48f89c265009c7e7a76e7a18921c5ef88f3504391)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs

- CloudHSM CLI (SHA256 checksum 2168377badf708047ee14d2bc57b9234f01ce2c9f42568dbd1185723a63eeeb8)

ARM64 アーキテクチャの RHEL 8 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 69db8b832edba06fe055ce9772bfad645a86e5e530d20e8f14738fc7cb06a5c7)
- OpenSSL Dynamic Engine (SHA256 checksum ac7ad7b11e63d1fb7af57ff1e61b421aa0fccbca99ed15b0ddec1602fea2acd3)
- JCE プロバイダー(SHA256 checksum 7520a95076a7a794219677489d2c764502a6dcfeea484f10059e283f1aee6975)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 596acc3164695a262b15edbb92eda89ffd4b8d06b91617ce91058c2d99ae2a53)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum bc3146d5dc7aabe5359c41d2bd0fd97ed4bc2f389be6a4b94423065af9d60bd0)
- OpenSSL Dynamic Engine (SHA256 checksum 05d6afa19949b69af4325f55d285a89255ab2c7487aba898c74cbd4ee8e30fdf)
- OpenSSL プロバイダー (SHA256 checksum 3deba48c356478fdbb0236ad435cee8f7f8cfe3cec1a894aea3f79d80d398cce)
- JCE プロバイダー(SHA256 checksum ebacb95d8f92528e236a5342cb5c1ecbf74c228428fb4a080c0c972b697292b5)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 575f54610d26561bb98b69548de2f61a17b5b93ed49be13821be536d76ffdb75)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b0c918575e611f486902ebbcc3ee84e15217247f5577b06764cfc620660c6ec4)

- OpenSSL Dynamic Engine (SHA256 checksum 57f0afb66c34a91e96bef4964dd3fb487fe5b9b06d0818ee9a2c90ad60068321)
- OpenSSL プロバイダー (SHA256 checksum dcf5128c6913e43e0a304d0221c243fd5228f235b78fac5ffd43478bb0d96062)
- JCE プロバイダー(SHA256 checksum 21d43992ace3aec9ec3711e7bd64ec716d3281f627e78a6f7a328a8e533beec4)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 096844ef80a7d573843682cfb3797a0c45608d004639ab0e671065f3964c06e3)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum ec8c275832f9936e698106454e930268d7f2f310faffa2b7d03c3c03c6e297e1)
- OpenSSL Dynamic Engine (SHA256 checksum 316c3929363ad1d764f356bbf56dff121b3bccefa5060b94c5d9d7955f828ff4)
- JCE プロバイダー(SHA256 checksum 4c53b61bcdada0a684b605c0630f52b6b3bf6e9ed64f4c623f35fa4283e9770d)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6b4ba5d3d06e5234b64b7c60310d5b477c2d8732fbbd3e9c60dcb56f2ef69a6e)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum d26d10425572f0624d2205e1409b88ccc5d793bcfeef6cd26fdcb5c41f5c8129)
- OpenSSL Dynamic Engine (SHA256 checksum e3b6cd3e93f2645d9a23add1a514e314ad3d48a93b253e6b984577b8e502e1f0)
- JCE プロバイダー(SHA256 checksum 7ab3ea04d34f92c585121b110f620b632616a5585a4ed153fbb61fedef667b06)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 9cf4a05ebf2bce4a0c9e57dd2eef87ba47ac05c4a303d4465cbf41536736c685)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- JCE プロバイダー(SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- JCE プロバイダー(SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- JCE プロバイダー(SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs

- CloudHSM CLI (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.17.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- JCE プロバイダー(SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
  - AWS CloudHSM (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfffb66754367883e2d7e997a363f30) 用の Javadocs
- CloudHSM CLI (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

クライアント SDK 5.17.1 にはバグ修正と改善が含まれています。

JCE

- クライアント SDK 5.17.1 は、OpenJDK 11 をサポートする最後のリリースです。

バージョン 5.17.0

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e0656d606c1f75b8ecf11df79f2d4aa17c0b68983bb20d271dabe02f7b97ca47)
- OpenSSL Dynamic Engine (SHA256 checksum b8917abe48799035018bc044fbe2e9db8d5568ebf29663ecc12d53ec2be5efd2)
- OpenSSL プロバイダー (SHA256 checksum 61e49cb0f1f9eb7bcf76f5ef36f01cf1fadfa56bf5eab94b05efdd93214c9cf7)
- JCE プロバイダー(SHA256 checksum 4ba2548fa27f3662c56fc140386366cfb40cde899695412c6d4731a7d0a70e43)

  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 0a9de06b3f15d38f3aa02dd0288d3a9ad41f525f692f539664168b9e6b95e18d)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 9bd40e13bf777a4651721ebe5613cf2e4f0f339679d50f06b81f7f1b73c1fef5)
- OpenSSL Dynamic Engine (SHA256 checksum e592419bc0449664f0707281a46ebb775091386ac5b297e35ff0e4160468cbe0)
- OpenSSL プロバイダー (SHA256 checksum 46358ef68388928702d780564811814aa826c143ccc0624a2a45d8995226d33e)
- JCE プロバイダー(SHA256 checksum db5bd1494c742a64ca3f2bc57253b17655ad38ecc27d45422965466c1caf740f)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum e3d21b4460a421058f65a597b0bb30a9fdda890363ebee107b124a1212d5a042)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 17d96d5f4ad25de74fadcc7272fac825d43c496b0b7912e4d32b1ccfb67250bc)
- OpenSSL Dynamic Engine (SHA256 checksum 3d3de61bd16ef849e53827709b8c8bd9eaf89bd1b7d56b1a02455bca38fe1eb6)
- JCE プロバイダー(SHA256 checksum 5c268313ebf16cf762619a1e75f1d44efdbd6eaf5ba4bbce67ecfe96bbed9775)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 43b678eff0225590cc59c02645332f4459c8161a6742e562eb95902d1bec9134)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7910ec55f8b106d804290b77e6d13cefb85ef79cbc728f69bb21eebeb7245c6b)
- OpenSSL Dynamic Engine (SHA256 checksum 399c16e0fbfa372710a83836517cfeaae0895513931427a458e7994ce91a321d)
- JCE プロバイダー(SHA256 checksum 1678581e31fcfde58625ca1e3c0762c4b96095065a868a5b7f0872214e5ab9e8)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 52257b580d73d902e9fd29f2c334f88cabee484c7eec8af28f7b2aa7864466e7)

RHEL 10 (10.0+)

x86_64 アーキテクチャの RHEL 10 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 037102ee2b7f9c244d647cfb06da282b99349e208d27b7b0aa80bd32e382a2d4)
- OpenSSL プロバイダー (SHA256 checksum 3d6d97df03a003bae579dabbf9abeefc1bd2fa6d799a6db01c3ef98d02145393)
- JCE プロバイダー(SHA256 checksum 4e69ec89acfefc19f922c68988d4674718d49d5cda2e0a288f7f0343ebfc270c)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 38a67191949cc8cb0c69ef2fe4694514b2ca6af4cf5f8763bd7f6860c501675a)

ARM64 アーキテクチャの RHEL 10 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum bc383aff236d4b2a0fdbc5fd81907cf9212ad8f3cfc91fadeb6cdc14b9d3d581)
- OpenSSL プロバイダー (SHA256 checksum d9ead84f8d2e6a09044646d62fed293ca082431e5e0338bcbbfcf8acd023ecc0)
- JCE プロバイダー(SHA256 checksum c61fdb5f05cf486619deb3e23ecb171ce4485bb64c891bf4480fbe6a4276c3bc)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum fbd4387ef3fab177de8c888d98d1b24bc042d23ea2f93834dfb5ee918518f406)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7fc68025493fc4a0bc4d3b2fd11b17da4481b6bee51fc86a40f95fe4c206aa60)
- OpenSSL Dynamic Engine (SHA256 checksum 3e786e9218f657c93e1bf1b54c81f0f573e39d10147cf99485da73677b0d67ae)
- OpenSSL プロバイダー (SHA256 checksum c52b366f9c6c960d697794bd7840888bc51e1ee5aa654b8d2c83ec1626797c56)
- JCE プロバイダー(SHA256 checksum 427e9c68815188b59a0f9b2fd14793791d9a4ce4d3e5ddc34fce916af18a45f1)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 49d128639eea0e588c95281635a33b0e62837cf5a9d992e0bacb5b598a0037ed)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 3ae8f5068abb3d9ca75ac21b160acc7ee84c7ded6107a889602448af51d5bd82)
- OpenSSL Dynamic Engine (SHA256 checksum 64c5feaf86fc40e28616156f17942cc46d8ac93babdd8af59dd98a1f2e73117d)
- OpenSSL プロバイダー (SHA256 checksum 24df4296ba0ae15f6b7060ccffeca0d1a24f58d9e8da8eb6a4afa3a6a50770fd)
- JCE プロバイダー(SHA256 checksum 648d7948ab8a1271d7cf0eddc2ca32e960979be481b99a1ce2ce8b9fa3a2b2ed)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum edd1f3707aeea285a55cf24d2e2385489458a69e5d7dab4014783f5cdc232d77)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 85e070ed3273ead9698711dfc2c3e590dda1a01f99074943459ddc5ac052872d)
- OpenSSL Dynamic Engine (SHA256 checksum d0c8e5603f4c99d0ba547c0ba40361c15977f3349ac120d514afed59abf32c83)

- JCE プロバイダー(SHA256 checksum 4567b9a236d0dfd0410863beaced6d5d4e71acb06b2b136c7c40ce6e704f3698)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum f8a090752849fac9bf99ef67b59f571e476908dccec1285b466c25e6cc71793c)

ARM64 アーキテクチャの RHEL 8 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 71ade413435947eb2093727cb0e7a5c5f084ca859cb38d728a8c3d23f3b49bd0)
- OpenSSL Dynamic Engine (SHA256 checksum 02210da06b3fd22813fce8b89b753d150c8f321df405c40dd9f0172dc03968a7)
- JCE プロバイダー(SHA256 checksum 80ac5dbaad1f644d6b9551e6a1d95d4b4a51ae82d34e45593be77b1068f7c910)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum c3c56c3b9fed1dc51761d9412007c727d3b608377a089bd637a44bc374744d5c)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b3869a6fe9f2031cb5f36ac8d2a5a5280451159e7a68f0d6f91f98b5f0100939)
- OpenSSL Dynamic Engine (SHA256 checksum 3e46443a3f8425f62cec8d8ca414c4b5446b86b42012877ebe44a17303403413)
- OpenSSL プロバイダー (SHA256 checksum 69c2e61894e954ac1d0a785025902af5aa1922a7f6011a847f2111e153db706e)
- JCE プロバイダー(SHA256 checksum 663da1a7b13a1055bb90fd7b168b69fcfb000eb91c1c3ce6062d8cc8db607fe4)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 0d57b4f08bd768ed595978d2a6d3fe6bbbf8398a784c66c4e30c19ecf6730b5d)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 94d9a36a3d3f2f04c10539713b98ddad68654c739572064e4c123005a9caf575)
- OpenSSL Dynamic Engine (SHA256 checksum a442950055a7051f50b4cdfa7ec347c1b207b0bbf7e159b31a3bf96896b66146)
- OpenSSL プロバイダー (SHA256 checksum dcc4b5a370da176a86e47683a78703a5604d6b7b49ba687377460f48c3ddff26)
- JCE プロバイダー(SHA256 checksum 1fe1fa59a3e5c32edd5668604e43f7ce152473514b25ee736dfb0e9cf87c92c9)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 2bf2c13bf1520eaf2a45eb55aaba3334fa54da686d22d1a03005ec8f62ea8895)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 2dc1f00952f69b59b0498e85c2f2009f80c9a2f9f28104995b0291da5f45d805)
- OpenSSL Dynamic Engine (SHA256 checksum df0e7ff8852a93fb4c5807317d3438ab96ffe69f19ba144d872991cbcb4172f6)
- JCE プロバイダー(SHA256 checksum 77a3cc37e066ae0ad5655ddc3dd38a0513bdcbd3c9d3d780e2973ac42a75035e)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 88ca1614178771991649b8117ddf27ffb99c6842ee89f68e77c6df3470e6ba78)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 0e1f62783c16debf9188c9cf0085fe972671e5ea9bf0fc2fc3255f908885949c)
- OpenSSL Dynamic Engine (SHA256 checksum 109d25583e2d5ed0fe821a77098d8f2c815011f7873168e0110aeb857dc903ab)
- JCE プロバイダー(SHA256 checksum 9151661a39f6394aa3ace682bd7964095ed5300244582d3f6b1f81255ebf22cf)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 66e408cea800a93becf378b8611595ad8cd1bf51e0ae7feebc548fe3099b477c)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- JCE プロバイダー(SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- JCE プロバイダー(SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
  - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- JCE プロバイダー(SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)

    - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.17.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- JCE プロバイダー(SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
    - AWS CloudHSM (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- キーストレージプロバイダー (KSP) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

クライアント SDK 5.17.0 では、複数のコンポーネントに新機能と改善が導入されています。

プラットフォームのサポート

- ARM64 アーキテクチャの RHEL 8 サポートを追加しました。

JCE

- すべてのプラットフォームで OpenJDK 25 のサポートを追加
- クライアント SDK 5.17.0 は、OpenJDK 8 をサポートする最後のリリースです。

新機能

- OpenSSL 3.2 以降の互換性を強化し、暗号化オペレーションを追加するために、新しい OpenSSL プロバイダー SDK を導入しました。

- EC キー生成に Ed25519 曲線のサポートを追加しました。
- CloudHSM CLI での EdDSA (Edwards-curve デジタル署名アルゴリズム) の署名と検証のサポートが追加されました。

CloudHSM CLI

- 非 FIPS hsm2m.medium タイプの EC キー生成に Ed25519 曲線のサポートを追加しました。
- CloudHSM CLI での EdDSA (Edwards-curve デジタル署名アルゴリズム) の署名と検証のサポートが追加されました。詳細については、「CloudHSM CLI で HashEdDSA メカニズムを使用して署名を生成する」および「CloudHSM CLI で HashEdDSA メカニズムで署名された署名を検証する」を参照してください。

OpenSSL プロバイダー

- 新しい OpenSSL プロバイダー SDK は、hsm2m.medium タイプの OpenSSL 3.2+ プロバイダーアーキテクチャとの統合を提供します。

バグ修正/機能向上

- 暗号化オペレーションの再試行メカニズムが改善されました。

バージョン 5.16.2

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e223649cbf689afaeca445b962cceaab3c143b901ee7974a7a5360b6424c409b)
- OpenSSL Dynamic Engine (SHA256 checksum 9b51e8b1429f22705399299c111385e9434873a1dd26dda2c70498646590148c)
- JCE プロバイダー(SHA256 checksum 45ec10c2e3d592fb1fe56cc9dee1d8653e3a6dfa15b8185618fd3cf66c567314)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 8134a0e316286310eda1606b6a14e1c9fd6f0e27934a43e55e24d3094ef870a6)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7556c69294d0c67b359cf9995ec7047cdc90e21542d2c9d5e110e35f13c329c5)
- OpenSSL Dynamic Engine (SHA256 checksum 6ab4f3b9731a6f26891658f5e3a0ff74255096390c42aa7f702ba07e0bd924b1)
- JCE プロバイダー(SHA256 checksum 98b80696762a73f3f9fbe6c3c7af0f8f9db23ba63d66d4d941690b6698019702)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 649397606f527f330049cf7adb5ad667126dfd7f63f878f1f46b469e2f127814)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 073ee736ad9c7de6e6451afdecc4d77e6ed1a63f314a7fc09373ff35986fa0fc)
- OpenSSL Dynamic Engine (SHA256 checksum b174fb24a15f1e22dd520ea9ac4ac14229c2adce14ac6df8901f9d3c0e7e24f6)
- JCE プロバイダー(SHA256 checksum f18e42f2679ab0fb51c421965a5dd6c8fe1361442d92403d1d230bedb003b27a)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 5a6c83833b60017dd494c31fd4abf3896f57177be7951674c85d0d4e909d2041)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum df2c97a27855916983f242eba05e0741335158432ac46ebbeae47c4a3b7b0bef)
- OpenSSL Dynamic Engine (SHA256 checksum d25456e3da39c63d9c0d4a928815f995239b146f4f0b26a8c50c742905543207)
- JCE プロバイダー(SHA256 checksum 7e31734efbaae49203e57d7ad4d1e291faa5b7e16c469a3596da82a7380bceeb)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs

- CloudHSM CLI (SHA256 checksum b77eb1f9ff58ccc864854d7f261e331ce50836cd770efc8e8eaebb33d78c23e2)

RHEL 10 (10.0+)

x86_64 アーキテクチャの RHEL 10 用のバージョン 5.16.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 8044cb40c9c65ae944900c740308e68eeb0156d3b93051d60463cd3f6095ef67)
- JCE プロバイダー(SHA256 checksum a1b7f110c2b15ba7ce9adee74baf1f108d85146269321991314f9153a074ca27)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 3ded29fdb74537b284cf871af91dd64fce8483770141a3e4470e94fb69524621)

ARM64 アーキテクチャの RHEL 10 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 643fa23ffea47dfb21fd326cacf99e0e45f44d767ca6198ab06023239eda5d2c)
- JCE プロバイダー(SHA256 checksum e6b5c2760c60f5c82a248d36f3b47cea9b1bbb743cf1c02a1c13554b70ab87f8)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum a6af281ff9f429f964ac9d2babc251b4423748d2b2e6da22fc51b9fd5c59714c)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.16.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e0a44849c9cb562cf803ff182e54a35e7eee682efa2a99fdfac51a996822d126)
- OpenSSL Dynamic Engine (SHA256 checksum 97463a8a88239d4a81854f2a4da7e0371dbcb375a662c7eae7753c371ef3c39f)
- JCE プロバイダー(SHA256 checksum 10cd4b9826451b7e9a11390cabcc7df548f5b3197e8bc87f5efe04605e503dc1)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6c7fc59dcce4d7bc0c4460ae7c793f13fcbb306924942bc6e1457f729602f76f)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 7b8d3f5e2bd09f30ddc1599619c63aa4452b16e1037bf10af070c055ae48cbd7)
- OpenSSL Dynamic Engine (SHA256 checksum 4edb0315c050b3b0699cb244fe36b987c9fd2921999cf7519fb669354d2efe2d)
- JCE プロバイダー(SHA256 checksum eae2064d9523df9f819ef4c7950a0ad15ef8bb01d8dcec062ee2558f5bebf029)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum f9f4073ebcde2b654b706efb1c0f99c08fe24903576e760029097b0e3c783f6a)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.16.2 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 9a3e023fd5ac8c444ef07bb28b96e81b467e066c9dc2da19f648f38eb04d05c9)
- OpenSSL Dynamic Engine (SHA256 checksum 6fc6ca58f16fe2f486f4730cd0a6029eeeebae98b3aed41fd8c605fe5c36c7a0)
- JCE プロバイダー(SHA256 checksum cb6e9eccfeebca7377b1fb815c2ae306b39b271513ce73a6cd5f1dccb55eac4c)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum e848f3f2221e7f6b984bd2001331f9e90f20e6d7351484e4e2f12431c00d87e5)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum d993f8c84d15cf93432b73cb37693c4ab68b3e765c35742adef3f947744b9893)
- OpenSSL Dynamic Engine (SHA256 checksum 055f0e4cc74ec7ead77fff3970422b4f1729e26a2cf3cc2be530f0e6a06f887e)
- JCE プロバイダー(SHA256 checksum 5bd62f8f5cb3ff4ae9430020aa4f5c8a29cb635674d2400d989f512c370c755c)

    - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum de263f6035cf7280a982434448aa079164050e4a09508816fd06bf000cf15f0c)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum a6496f28f82f3e92b0acc8e3ae1fbe230a6d3ed82586e8ca6fe6d2f82c275e7c)
- OpenSSL Dynamic Engine (SHA256 checksum 82275d6ac2628b4291162e3609d6788b4cf01d6488a9b6a84ed8787a5a96e873)
- JCE プロバイダー(SHA256 checksum 67800957c69efca0fe01623364cb3b152f2780d15624955130d507a3f591e8e5)
    - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 088f53d75250aa0ce3db7a7ba3f06ad7804b3dbeed6226683a3252890a867ac6)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b9799ba75fb8885782283abae7f49aa76a72d27f692654dd3a415157f0ea9043)
- OpenSSL Dynamic Engine (SHA256 checksum 3c7675e9b7fa4bb692865559f09fc1f9fd78dadb97d8bd6b5d1fc03367d75b86)
- JCE プロバイダー(SHA256 checksum 37c21821ad4d9fde9551112e8a6d7ec6b7bb62a540c0c08b6edd650d5c87b5cf)
    - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 7d3a724892e993aebfb06ad5c3d1ef8fc650d65dff0a7de3cb9c17c089badc58)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 8a2b392e48b7f9021966f6c457444ddd0964f9bb85b7ecb988bc0d372bbef6c9)

- OpenSSL Dynamic Engine (SHA256 checksum cca0736d208d59d070ced8b82181e24919f21fbf0497e5a8a2f40436e06a8dd9)
- JCE プロバイダー(SHA256 checksum c12e1b9bfe1cc78ecf35892050dd06904c6a29736cc628ac1481dca4ec3e3bfe)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 586ee7663d54e76652e9f073772fcbda218dc3c6432a221d664ffbabee3590cd)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- JCE プロバイダー(SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- キーストレージプロバイダー (KSP) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- JCE プロバイダー(SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- キーストレージプロバイダー (KSP) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- JCE プロバイダー(SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- キーストレージプロバイダー (KSP) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.16.2 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- JCE プロバイダー(SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
  - AWS CloudHSM (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d) 用の Javadocs
- CloudHSM CLI (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- キーストレージプロバイダー (KSP) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

クライアント SDK 5.16.2 では、RHEL 10 プラットフォームのサポートが追加され、Ubuntu 20.04 LTS のサポートが削除されました。また、クライアント SDK 5.16.2 では、新機能を含む JCE プロバイダーおよび KSP のサポートが追加され、その他の SDK に対してもバグ修正や改善が行われています。

プラットフォームのサポート

- x86 および ARM アーキテクチャ向けに RHEL 10 のサポートが追加されました。
  - OpenSSL Engine は、OpenSSL 3.0 で導入された新しいプロバイダーベースのアーキテクチャへの Red Hat の移行に伴い、RHEL 10 ではサポートされていません。

CloudHSM CLI

- インタラクティブモードで事前ハッシュ化されたデータに署名する際に、トークンがクリアされない不具合を修正しました。
- 3 つ以上のトークンが存在する場合にエラーを引き起こしていた、`quorum token-sign list` に関する不具合を修正しました。

JCE プロバイダー

- hsm2m.medium タイプ向けに、ECParameters および ECPoint を使用した楕円曲線 (EC) キーの検索サポートが追加されました。
- X9.63 KDF を使用する ECDH に `sharedInfo` パラメータのサポートが追加されました。

PKCS #11 ライブラリ

- hsm2m.medium タイプ向けに、ECParameters および ECPoint を使用した楕円曲線 (EC) キーの検索サポートが追加されました。

バグ修正/機能向上

- 1 回のオペレーションでより多くのトークンを表示できるよう、トークン一覧機能が強化されました。
- 事前ハッシュ化されたデータに署名する際、CloudHSM CLI のインタラクティブモードでキー使用トークンがクリアされない問題を修正しました。
- hsm2m.medium 上で一部の HSM スロットル済みオペレーションが自動再試行されない既知の問題を解決しました。詳細については、問題: バックアップの作成中にオペレーションが失敗することがある を参照してください。

バージョン 5.16.1

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 342e81846436708cfc1fb459a7ef1c7b065d8a68b50a5e63653c24918d0338a0)

- OpenSSL Dynamic Engine (SHA256 checksum 4e83bebcd20201c04629a03b70136df8d225d3056346789054c4e2ce8c9b3cba)
- JCE プロバイダー(SHA256 checksum 014e3c804a56087f8855ed30e480f9c193def0e1132a4769eae3ecd235d76c4f)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 854c5c21f9fbef4f53a710340f213c272365c2bb1233106266eca5a5aa547e11)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum dbe9498d91e1e2e8f80fc870a4a6576b0dc801dc20cb7a82bfded79b3a362ac0)
- OpenSSL Dynamic Engine (SHA256 checksum 76cff941f36275163db146e05bf5fe64440248643ceb5dfcb3b64103f4f016a7)
- JCE プロバイダー(SHA256 checksum 8c38082797172b5630c74a8fc0c9e2652f0898bd2232b72911fbbaaa0260b5a6)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 2a0640e32405a7db138afb2444a542eab11c0a7d9610d5406478c7ac4602a14b)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum cfa59458ec239553c86ac55773bebc69a5dc7cdc08a3927917fd5918e10abe93)
- OpenSSL Dynamic Engine (SHA256 checksum 49e50bece7d73f3b7dc95eed1df856a6dee40e27f24f40f015c6a4a2e8dee839)
- JCE プロバイダー(SHA256 checksum 85c8853860aa36a6c54e75c94d607a334d64dd34683ba70430ff5be61eedcc56)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 0efc3783e8429331aa446e45776fef46c7f9726a4768763ba4881dc2b05e090e)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum dad60a0380ef0ea9c469cae4de10dc47124bbecedafb6956d009734cd49abceb)
- OpenSSL Dynamic Engine (SHA256 checksum 7ee303421d94544cbe9df03022c48af327833e631d8f1ec59a466673b6e9395d)
- JCE プロバイダー(SHA256 checksum 5ebe8157abb042ee92a7edfbfcb98bf79fc3a9907684565176fd28c387c08e88)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 5e76b4a9021a3c92e59fc608c8263af731013835738977f376fb8ad9189add56)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.16.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 696e58f74d89bd8e39f85bddc547f8c94fa2fbca498318a7bb94f22e8be7a668)
- OpenSSL Dynamic Engine (SHA256 checksum 86197cd8bdd70db91331bb8380ea094352b4087c95a04768d2cefc3bba18dffa)
- JCE プロバイダー(SHA256 checksum 37ccc2df3e8aaddda74dc060f7f5bbe63e4769311f2ccbf313a8ab5d8831f206)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 45a6db22db9ae64c239b49ea9281d29d2b4364dd63ee2408f463de17c4da1877)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum fd79fbccf504f53a6e70497135fcea4e20982fca75b628124c7510dcbaddf56c)
- OpenSSL Dynamic Engine (SHA256 checksum 23fe7407be9c5f8da0ae64e560a9741887bc31936f0b88c8d1490e3c6893a8a8)
- JCE プロバイダー(SHA256 checksum bd1c43e3e28c2f71e78a41342c064ff873fd8e052f89f90227fe00eaa28f089e)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 3e877a232303052b8a6b2b869dcf228edbb0da913d3d41393f622831f4455a27)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.16.1 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum d396c53c229b1eaa7cb30d4fcd17addc9170c7942795d1c82b23a157ec379686)
- OpenSSL Dynamic Engine (SHA256 checksum c01f2cb66a6c5be839906b25c6a7f7990507b1b8ec3da34c128c1a90838e48df)
- JCE プロバイダー(SHA256 checksum 1745ab78de0712111659e91301f80e771e1b7ebc2ff8c4cc5e37bce591058520)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b3ca40ba66062856ef63848c71f6e9dfa0a46a2b18d44c2d96a798fc4a4fb9cf)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum ade113ee72547cb6a8fe91a4f3ac403f462d0acd0a306f01d16ac90699f9b59c)
- OpenSSL Dynamic Engine (SHA256 checksum 9e00cb32b16fbb286a853f65c5c5154c3e805ad67820d829a0c5343f12cc9e00)
- JCE プロバイダー(SHA256 checksum 4fab322a138e14372aba394de29f444af640479dd338539be84e0cf659c8993b)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum f8c1e369885eb77ba4517388dab47c067a617e860fb67a028a1ebde0f96acaef)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 5e15a69f8fb1429ed5c10429c91323fe82a63c0fe7bcc2f968820733d1737449)
- OpenSSL Dynamic Engine (SHA256 checksum f5136ee61bd34b74d59a01d37964b70a0f80ec5981f732b265af1c5466309e6f)
- JCE プロバイダー(SHA256 checksum acf5bebe7009461cdc1e31b95caadcbf80da09ec78d8f69142072da9baf61b79)

    - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 511376d2faf9f991f2193d08f80fc1edd53bf06daaa6fd8bd7a3fba0d6563ad1)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e1d787f10ee51d94732732811e4d2110f1b8e448e67fd47df8b53a2f7e56e3c8)
- OpenSSL Dynamic Engine (SHA256 checksum 29a9c41379754ce098a025feadc026f2f75a8638981f5b95ed07007d5b3d8510)
- JCE プロバイダー(SHA256 checksum 390d21e499ba181dc2774c3140433f6e60b49d9de3a892a89038c27e1aff157b)
    - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum cadadafaa9464f944e5c2446df92fb1777a9e1e5116cab1b359a8aa101cf7ae8)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 3b6a3d50b4a55c150f12308ee3914451897bc4cc4c8420accf1d2d706316fcce)
- OpenSSL Dynamic Engine (SHA256 checksum c366a3c5faf4de32a6dedc4613234ba3d331b0abac3241bed7d25e0109a44f64)
- JCE プロバイダー(SHA256 checksum 705416b47bc62781b5a6dfc78dd02dadb4f18f842b569c3e210b9535813b9e70)
    - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 811b501615cf34665b70f103905094ca84ed7c126f72d91b040aacda99dbf22b)

Ubuntu 20.04 LTS

x86_64 アーキテクチャの Ubuntu 20.04 LTS 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 6a2c33c78ada33fb435bbf8939b7cde3efe968e9f03250083dc6024b7ebd45b8)
- OpenSSL Dynamic Engine (SHA256 checksum 6ee178454e78d88e0ab92cd7c5b056d0c04cd6de192aea731d7ebcbd4c5ed761)
- JCE プロバイダー(SHA256 checksum 1e61ccd8d2b37c64467051d26e8d8bd592465ecef04282d6f4a9491707ba059d)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 645693f2ede4fa0d7879eedaa41ae23120d902f24c9a637ba088f277e703cb96)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- JCE プロバイダー(SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- キーストレージプロバイダー (KSP) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- JCE プロバイダー(SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)

- キーストレージプロバイダー (KSP) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- JCE プロバイダー(SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- キーストレージプロバイダー (KSP) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.16.1 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- JCE プロバイダー(SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
  - AWS CloudHSM (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- キーストレージプロバイダー (KSP) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

クライアント SDK 5.16.1 では、CloudHSM CLI において事前ハッシュ化されたデータの署名および検証がサポートされました。

プラットフォームのサポート

- SDK 5.16.1 は、Ubuntu 20.04 LTS プラットフォームをサポートする最後のリリースです。詳細については、「Ubuntu のウェブサイト」を参照してください。

CloudHSM CLI

- CloudHSM CLI において、事前ハッシュ化されたデータの署名および検証のサポートが追加されました。詳細については、「CloudHSM CLI の暗号化署名カテゴリ」および「CloudHSM CLI の暗号化検証カテゴリ」を参照してください。

JCE プロバイダー

- AES/CBC/Pkcs5Padding の暗号化モードが更新され、初期化ベクトル (IV) が null の場合にランダムな IV を自動生成するようになりました。以前は、IV が null の場合はオペレーションが失敗していました。復号オペレーションでは、引き続き明示的な IV の指定が必須です。

バグ修正/機能向上

- JCE における連続したダイジェスト更新オペレーションのレイテンシーが短縮されました。
- Windows Server 上で実行する際、generate key-reference コマンドが属性 ID の値に基づいて KSP のキー参照ファイルに正しい名前を付けるよう更新されました。詳細については、「KSP キー参照を生成します (Windows)」を参照してください。

バージョン 5.16.0

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum d63271304f32f49838390a58d94a2140ae8a744ac64efcf0e6a65983c858d862)
- OpenSSL Dynamic Engine (SHA256 checksum 25d78c1df82355601ed6887bb47d64a06380d001da15a070cfa89dce65417fb6)
- JCE プロバイダー(SHA256 checksum fcd876813e5d114e2b042765ff98b256410e988a4e373ea1aae6f260aebdd554)

  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 7f1441eb08daa7ceab91161900d89974ee86e0a3c54d878a29e43db775bd04f5)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 0b42e4e89cddda10c5be5f295a52d7a541bba057a3a07d3d9b192cf9eb49c776)
- OpenSSL Dynamic Engine (SHA256 checksum e8ee949d44b9f9ba64ed36eb7944acacbd1d76d7cf09d5492deae5928a9953e9)
- JCE プロバイダー(SHA256 checksum 67a6fd8f7b5dfc60ec6d6d380c31035675373b542205ec7e5a687175191cd275)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 1e6c3053b19aded54045a3baa9fe2545f81a35332366cf2d77dca92ee888e654)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 5a89ff9801f89f51e27e70869f2713cfe9bbd87d1198246cffaafe74aff9e809)
- OpenSSL Dynamic Engine (SHA256 checksum 991b6288289d07972915aa0be6dfd6b3c33c1d6312bc304225715e30832e688f)
- JCE プロバイダー(SHA256 checksum c5e673f79c2efd83195a288a5c8860c4bf74dba5bc8e422741dd5cc1be230fb3)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b9021b0b348c08f433e7cce247ab1710e2a79f33a82ac36e1d6a4672171028db)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 97677a7f9759637b548205d31c9a08a3e7b27dfa5a38e9a4e8e2398ae9869a5a)

- OpenSSL Dynamic Engine (SHA256 checksum dde369dc4efd9f59e5c9a5459a532ad5d273f87839fba6dd5dcde2c3a2b61517)
- JCE プロバイダー(SHA256 checksum b7d96e7453e26ff07ccd9437b102f4c21857c79daa6315054728fc7ac66150bc)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum c7386ac016b14ea8cb96a6ed655a67f46826cdb925bd87902277b5ce9b593d51)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.16.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 178d3e1ff0ec8cdc7f8c8be9aab772b4195602bb1c48dea692b628689be5ea3d)
- OpenSSL Dynamic Engine (SHA256 checksum 0488ad66cd825fafee70d1a1ccb1c8045b089af7b7d6ccadbec75f535376048c)
- JCE プロバイダー(SHA256 checksum cc9a60fa981a46c674de1e715d92ec1b8cd5c801394bfb3f0d101880589d4202)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum cb7e3ce6704f015d5b9462f12e6ad5489366fb3e80f554a2bdf7f04e688be1a9)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 9567fb016b641821869919d4d6e9754ae08fd549381446640ff135a1e225b809)
- OpenSSL Dynamic Engine (SHA256 checksum 8c46a9dd908e77a0d8412e7050621d0c570e091845480815c230e26d1917227b)
- JCE プロバイダー(SHA256 checksum 78a7c714aedd068e5e2f12f5044a6ce1618e6dfacbcdd91b4a9b505842dcc6c9)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d2340ef4092ca6c1fa1ac8b484550508a5d9d8670e663bef6e595a07f8151be3)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.16.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 47bf628a4a2663b8e8c7207b43e5af1d05ed2247e85e1371773596c132c9a4ec)
- OpenSSL Dynamic Engine (SHA256 checksum 49dd44243863121f7781e0f4ac556828c746d85db723864ccf16ea5fb042846a)
- JCE プロバイダー(SHA256 checksum c5e8a6664e6a8e5338a98b3845bed029f5936b517e6043aeb65ad58c3364eed8)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 71f2cd6bd8892ccaea845bfe575cdf20016dd7176e5216b76290e499c822a2f2)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum fd7e78050aba017b7dc859f0b1e6f354f56b22091cb262b328aad204064a2960)
- OpenSSL Dynamic Engine (SHA256 checksum 9d2a9e8049caa12c8027ad2963fa30f22e1032d4025e39a4e2d5985073c283f5)
- JCE プロバイダー(SHA256 checksum ae19c13f55e01538e989e8693aaf3a5d998c60f1ed71e721a34404a228947c20)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 7ebedb3e8c4b8e01cc1e5dfe32ae18c9542d9c2940c77bd71ee102093aaf1d1c)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum cfd9526cf495e81a2477b464ca3c124a89d07b1a366a14b39eeb27d6b5e2404b)
- OpenSSL Dynamic Engine (SHA256 checksum 02a5cc2cd9442b7bf982bb3ae44c9b134758b46e68e46fe4729b4adf1320d75c)
- JCE プロバイダー(SHA256 checksum 124d04f667a7222d624f528404a289405669f4e4a1983b8a7926758bf67e35ec)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b874556b34ff5f0d082ca19f62822cf97d6e7074ce3861d7577090f99cba0cbd)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e36d9bf900e195e417db261034d820cb935bec5135a0aa332d46019c70257cac)
- OpenSSL Dynamic Engine (SHA256 checksum 2c39129a5acce96b8693ce8c4b4aa174d9c13bc8ea3c2652fa505b4e0a933842)
- JCE プロバイダー(SHA256 checksum 03b70d834985a2500b73eecd831bff4f30389eb67460bf1d799466e59a9f9cfd)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 3fc4fc662188ebbd430d9b5e01a57774567b926f3478d0a455fee4968c47460d)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum c835d9bb604b5c47a702fd45267ffb9dfa37d68dd0734391f060835ea680a169)
- OpenSSL Dynamic Engine (SHA256 checksum 943213a0f14b50fbdf3a90286e7f7ab219ab77750698f5062013873419484a04)
- JCE プロバイダー(SHA256 checksum 0ecc3e65a3b94c2dd3913f44aba58764e5558a8bedd3b742a42e4769533f3734)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 8558328b224b850cf9e1b3c4f8773565a90e38ab43e2c9e90bc5b841f98898d2)

Ubuntu 20.04 LTS

x86_64 アーキテクチャの Ubuntu 20.04 LTS 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 64a01e84d898aca3cd03f22022cf8dbbc806a1735a84df5f820ba5bfa3339b0e)
- OpenSSL Dynamic Engine (SHA256 checksum 53a73cfc60c3fddd3ad9173bf6e602faf289f20935b3c469a137847148442954)
- JCE プロバイダー(SHA256 checksum 09544cb8129af4d06a241e1fbc0b6e6c207e1a518e5cb202f7b6c9802507d1b8)

  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum e1fdf1e014541f57d3ba4688bad88e7557e32ba974e020b4da76f85f1fc6aa29)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- JCE プロバイダー(SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- キーストレージプロバイダー (KSP) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- JCE プロバイダー(SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- キーストレージプロバイダー (KSP) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- JCE プロバイダー(SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- キーストレージプロバイダー (KSP) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.16.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- JCE プロバイダー(SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
  - AWS CloudHSM (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- キーストレージプロバイダー (KSP) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

クライアント SDK 5.16 では、JCE プロバイダーおよび PKCS #11 ライブラリ向けに、hsm2m.medium クラスタータイプでの ECDH (X963 KDF 使用) がサポートされます。クライアント SDK 5.16 では、CloudHSM CLI を使用して Windows Server 上で KSP キー参照ファイルを生成する機能も追加されます。

CloudHSM CLI

- Windows Server で KSP キー参照ファイルを生成する機能が追加されました。詳細については、「KSP キー参照を生成します (Windows)」を参照してください。

JCE プロバイダー

- hsm2m.medium クラスタータイプで ECDH (X963 KDF 使用) のサポートが追加されました。AWS CloudHSM クライアント SDK 5 の JCE プロバイダーでサポートされているメカニズム を参照してください。

PKCS #11 ライブラリ

- hsm2m.medium クラスタータイプで ECDH (X963 KDF 使用) のサポートが追加されました。AWS CloudHSM クライアント SDK 5 の PKCS #11 ライブラリでサポートされているメカニズム を参照してください。

バグ修正/機能向上

- 切断時にセッションキーが正しくクリアされない不具合を修正しました。
- hsm2m.medium において、mTLS のログアウトオペレーションが成功しているにもかかわらず誤ってエラー応答を返す問題を修正しました。
- Windows 起動時に接続が失敗する問題を修正しました。
- Ubuntu システムで SDK 情報ツールが誤った出力を表示する問題を修正しました。
- FIPS モードで、より弱いハッシュを使用した ECDSA が許可されてしまっていた問題を修正しました。

バージョン 5.15.0

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 41ef3178811df1dbb03b2cbac83fe0f4768bdc9b17005c409f1c0229f93ef11c)
- OpenSSL Dynamic Engine (SHA256 checksum afa1f9f8bd99f54866dea1b8928c00b951a6e492f5f36d0d6c7c38fff341d609)
- JCE プロバイダー(SHA256 checksum 6ea775e05570ef3497a4df5c35a6ec1c682aea73c48e7fecec3e541af995759e)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 78c10bb213dd14fcfc5836e358de6aedac61db05125fd61137b0082214fdecbe)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum cd0617d29b6d64c00c8e14fd9a92f604e14acda4746f4f0b86b9de42367192fb)
- OpenSSL Dynamic Engine (SHA256 checksum bc9acfdd04eb1246eb3d5b0a8f3736ec017c0d1699d5395f85868d4a1722cd83)
- JCE プロバイダー(SHA256 checksum 96d05301486206577b7be05aac561649167b57272c9c06ff839fd8ff2b5d96d5)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum b870eadb27736a2cde98022d57e9704c67ae15878cf0b910738859cdabaa35a2)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum c70ae4f0181a8187c9380481c51c1d03e12236dd86863ec818ed3f210b294c8e)
- OpenSSL Dynamic Engine (SHA256 checksum 08e9fd1dd80efa637f9a1727bb0de205ba124a3776b2e8bc21008ee458063a42)
- JCE プロバイダー(SHA256 checksum 7932ed060e72c53b2556f30694b0ffe5342b244b6628c7a9dc03966aa49c8fe6)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 0a939e4d5d0a2ff308c7a1d9e73ebc865e426214d556bde1bd29dbe807fbb583)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum bcf907a8fb4722b09c54b8e5785fe2b8cffcedd6ee3fdda2d21879a012138077)
- OpenSSL Dynamic Engine (SHA256 checksum 8f70edc3a6a4a1bf0264c6567b1dda1ac69055f206753f88eeadbb8bf3bf9f38)
- JCE プロバイダー(SHA256 checksum 75dd67736bb08fe7e46e113af10803a255ba8edee3016ca963c1ee94fe59d43b)

    - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 1c2ecf90c955281d99bcd8d1956d63debb15bbc8419744c83f88821ef8b78aee)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.15.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 65bd0b815eebc806674a7bf7c54e9f884595881547f5fffd08ff6a38aabdccbe)
- OpenSSL Dynamic Engine (SHA256 checksum f2af9f5882ab2e5a11defecc660f8af5c4d9d6e2e2b89873e6833fc2976f44ac)
- JCE プロバイダー(SHA256 checksum 5124bace5d1544775c891f13a0e309b30dd73699116c46ecc9a77bba5f9cf633)
    - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 1eec31fb2c0ab3c2839d9bdb37874cf8dbc74a383279068fce9e8613966d06e0)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum f498697519afa04b6b0362a4388a6a38e2cb6813b781d6ce3d97a7de89c5cbfe)
- OpenSSL Dynamic Engine (SHA256 checksum 5410da63108a1b209e567e9bccf8bd7e4035af88b0d58b9d78b10917be1b40c1)
- JCE プロバイダー(SHA256 checksum b46f233e6994d2c0ed505dc5c717ee3009daaeb2063d260aa78e06273770bffa)
    - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum e9f93eaa58db2f7ea1174164ef96ab219700933d353243c3c6ab1aebac5ccffe)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.15.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 87131e179d0e60ade302ec07b22803cfb39294bf060b786c41f154d95791ac94)

- OpenSSL Dynamic Engine (SHA256 checksum f412a2f5cd761db5940288bb252ce060d44735c6b436bb6d4fa7d3687a44a026)
- JCE プロバイダー(SHA256 checksum d0844f55c08f9ff9c393138a9041efe1b59dc3dce20c0b2c23406efe6acc43db)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 9fd8033e478ce6d7d640c063c4f007359cb04c19d519826a745ad0885f96a0f8)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum ca5f2f80ae921cfebdc5c8bc35c39d2b19cfabfd5981932409eaf2e7c00a9097)
- OpenSSL Dynamic Engine (SHA256 checksum e44cd7b678a421957c84e4fc0f70280360fd4e1e66f4cabdd1b20b955ee5fcca)
- JCE プロバイダー(SHA256 checksum bc6382f12769c3d87c34522d0d616b7a8c108574c003814e1300938c386655ac)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum f786be86c204680e670f7817376ff376733ebd96b51c31e9c98213078596637f)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 1ecf057d67137c0e9e1bb0dda57d581690c647b0360c3a617a8dc668919d08de)
- OpenSSL Dynamic Engine (SHA256 checksum 4f3d23e6b798f88be587108d8e6a225d796a3d080aef61ea384eb74a1270612a)
- JCE プロバイダー(SHA256 checksum b9fd16bdcc1fcf59fde0d3e0debee500b0b7edfdff69209e84d14393097fe9d2)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum a95922d9b44e64a64723db0c21ac89566515a5a6c87de990af4a9e1f40c7424f)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 966be12eb32de813ca07e766abf7b5616c0d2e105e9296d920aadaca10e5afdf)
- OpenSSL Dynamic Engine (SHA256 checksum f2840151d87b7f9cbff68993c25397afd48a16f054abf0f2fd4624662d3087d6)
- JCE プロバイダー(SHA256 checksum 01aebdda05640e50a82cae04c5cb6d33ab909dadd917bd834957d4f75ad8c577)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 2e5eba9409abc429828505779c512cd2424719766c0e488f50aee288966cf61d)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 614f74f101c9c64fff515128c5a59fa23de5047494fe248e8aabdab441092b3e)
- OpenSSL Dynamic Engine (SHA256 checksum c15d6db77b76bce690749b73b947567eb5f2d76669887843116d0ce56c1f8ea7)
- JCE プロバイダー(SHA256 checksum bca8511d5c0a173b0fef326016ce5091b6e6829fa2b3cb45ed5621290ae3e42a)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 2f5b673148b682d7e34619c51b5e8799abe7dc7fd4f046158a0d05320ba24dc1)

Ubuntu 20.04 LTS

x86_64 アーキテクチャの Ubuntu 20.04 LTS 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 86c8394b5ddff91a71194fb87c327efde36baa2380e559c04f9d543a6e74d61b)
- OpenSSL Dynamic Engine (SHA256 checksum 0a4227389fea61e6e7ac7cfa715eb341f7a4eeae9ed10e4c96da2c0dd4a18f9e)
- JCE プロバイダー(SHA256 checksum 65b2d926ff9dfbe6c7864bc3a41b3da2383bc731dd199bcef8805a0543fbe612)

    - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 09c2e55bcf72f9e530717950d8c5fdfd48574ae6ccb09a049526ca5b2a3b8aa9)

Windows Server 2025

x86_64 アーキテクチャの Windows Server 2025 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- JCE プロバイダー(SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
    - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- キーストレージプロバイダー (KSP) (SHA256 checksum 52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- JCE プロバイダー(SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
    - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- キーストレージプロバイダー (KSP) (SHA256 checksum 52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- JCE プロバイダー(SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- キーストレージプロバイダー (KSP) (SHA256 checksum 52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.15.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- JCE プロバイダー(SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
  - AWS CloudHSM (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- キーストレージプロバイダー (KSP) (SHA256 checksum 52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

クライアント SDK 5.15 では、CloudHSM CLI を使用して、クローンされたクラスター間でユーザーをレプリケートする機能が追加されました。クライアント SDK 5.15 では、Windows Server 2025 向けに、PKCS #11 ライブラリ、JCE プロバイダー、CloudHSM CLI、およびキーストレージプロバイダー (KSP) のインストールパッケージも追加されています。

プラットフォームのサポート

- PKCS #11 ライブラリ、JCE プロバイダー、CloudHSM CLI、およびキーストレージプロバイダー (KSP) について、Windows Server 2025 のサポートが追加されました。

CloudHSM CLI

- 次のコマンドが追加されました。

- CloudHSM CLI でユーザーをレプリケートする

バージョン 5.14.0

Amazon Linux 2023

x86_64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 05e7a3882166c694a7a09bc735f08f91c8145a4215176665eacacdf3e509abe8)
- OpenSSL Dynamic Engine (SHA256 checksum f4dd9966988418e100c276dc0d521f91afdfc0e6c008dbf8eda446ebaca83c14)
- JCE プロバイダー(SHA256 checksum c4dee5c1173f6a1c7683aedb58e61d329c36933435c416f288d94bc9a68a6b31)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 1fad069fde305450254287e43750d597db9af0cb8fd168300cd5eaed9e2af33a)

ARM64 アーキテクチャの Amazon Linux 2023 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 9f34163d02bce26c1280e589310cda891d27995c50cec0a7fe083100ecff2b69)
- OpenSSL Dynamic Engine (SHA256 checksum 7444a4daad6e4715c82d6c39a7b03a07ee0201a13fe0f98da96acbf9d24abf6c)
- JCE プロバイダー(SHA256 checksum 3393fe3a0f5c3a9c92106d74b7de074c818e095f97d2cfd600dbd47779b90b37)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum d9a2edce48c5f6646d5a351cb431712f2d2fc62d21f8318e7bb1ce579819d7f4)

Amazon Linux 2

x86_64 アーキテクチャの Amazon Linux 2 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 9c47b90bfa0ad51627cdb0dd8f148a56090fbdeb2490f1ab4009170c7b9c1120)
- OpenSSL Dynamic Engine (SHA256 checksum 215f9768331565085a317585b3dbe0514b251fdc428c96ed32491c4abb9fea56)
- JCE プロバイダー(SHA256 checksum a802f941e95fcbf0ef37775fc096c0d6ae4c916fa08330a9d56defc5f99ff2b7)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum ac10f23dd81264b1d10a0760f62b5006d2b2b66bf1c6378248ca9326afb65a83)

ARM64 アーキテクチャの Amazon Linux 2 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum f2d1550f043c655ef5d51e2ca14a0416886d99902e0702bd7f30f93b2c563d4d)
- OpenSSL Dynamic Engine (SHA256 checksum 2c8e2c81af53ba3646d1f947894d84b9b54780bbca79b58f354125e4ac9427c0)
- JCE プロバイダー(SHA256 checksum 3f9c881056e6905d46358585db72143b59971958a708fa4ac75cb53994487213)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 6d5985e44c9852409dd3d342239fbbf7d0f9ad43d445e0980153c9b9eafe2b6f)

RHEL 9 (9.2+)

x86_64 アーキテクチャの RHEL 9 用のバージョン 5.14.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum c4812210421ff2fa5dac8477a2e6b10552aabd88b1f03d717e044e7293823158)
- OpenSSL Dynamic Engine (SHA256 checksum 3f6aeaa6ae1faae7d8bba1596f358cad1eec9e562cf08aaab9ded92cabe94719)
- JCE プロバイダー(SHA256 checksum bd8f120b08f738ad4d1534b0e32aad903758d5e75d3aba7cb9ff6a77dec533db)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 84f13a9babd767edf90dabb0b14034ec5b2208898123a13966dd6b8519961c27)

ARM64 アーキテクチャの RHEL 9 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 3faa1de563aa9773939cab7b39907dcb7b981ed7225019f9070a9dd52db7ae70)
- OpenSSL Dynamic Engine (SHA256 checksum 13b41ac47ef7ee7bf78f585b8347ca4da9ebc296e4fc1e6a0c2ff5b333354ca6)
- JCE プロバイダー(SHA256 checksum 06803655ebe54d59c180bb17ac6fe56337bdd58ad0f6fe87c50ff8df32f70258)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum e6b4e9688d0db9d72bbee3450fe19736d640c9931adbd4f4ef73cb7ac2a08cf4)

RHEL 8 (8.3+)

x86_64 アーキテクチャの RHEL 8 用のバージョン 5.14.0 ソフトウェアをダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum e400aeea6dbf7721f97e71c643a0db4f5f1094fb197fc46dc0ab293de9d16f2d)
- OpenSSL Dynamic Engine (SHA256 checksum bfcc27d251e62f9eba0fd508e7d08dc62126642d4cdd0b5566183957768b8c54)
- JCE プロバイダー(SHA256 checksum b5500031b572c918a8df4a0347e01c8ea00a7366b865b310bd92427fa1ed53e3)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 000712d0a691efc64a6c5d54bfeb1ab48b315ebb5dd6926b0502e326bf700291)

Ubuntu 24.04 LTS

x86_64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum bbec70a198a4b173620b4018accc297ce6a6a80d372706e0101997d6bca35bca)
- OpenSSL Dynamic Engine (SHA256 checksum fcb77f75bd465b22401a09a20c4109858333402951012 63b7171cdcc4ac9f980)
- JCE プロバイダー(SHA256 checksum e1fa16aae2f6095c89a8bf392ef2cb9ca4db8853c858904ff90abf4bb491d74b)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs

- CloudHSM CLI (SHA256 checksum af32e0ac1c5f8c7f8cb40d255abe7c63ce9d981293187a3fb452562fa05756f4)

ARM64 アーキテクチャの Ubuntu 24.04 LTS 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 1232318b084347889f92136536537ca519373683ce39bcec20f6ac3fa2f42d7c)
- OpenSSL Dynamic Engine (SHA256 checksum 1745ee3a33d8e6ea72644e903c55c6a206204cf0a8bea200bc4a7b15736ed801)
- JCE プロバイダー(SHA256 checksum 7a44acabbc90c996594ed53661937f9242850823347f7c386f02fc041a97471a)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum e6778bd12c55fd152b50033833531fe569472f4f2bd9927a345eb126e8305739)

Ubuntu 22.04 LTS

x86_64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 6b4b1620e9a85267950633b171dd188b7ac7094e371e188fabc1bef7a911a16f)
- OpenSSL Dynamic Engine (SHA256 checksum fee5f0a65fab0f46ad58689af5dc510721581f31364d3be5cbbf79f5d9a60db8)
- JCE プロバイダー(SHA256 checksum 5883a3d15e160d65f8e26f185e1ee30f68becad5f6fcd16abc1c4586689800b5)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 7b1a8cd2962f4be7325154c080dabfb820beac5629e272c3a3ebc0e6cab11d27)

ARM64 アーキテクチャの Ubuntu 22.04 LTS 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum 8680f63fd74272ea0e4d4c32dfb96b3eb9b60e03483c202d4e9b65ee101a178f)
- OpenSSL Dynamic Engine (SHA256 checksum c757304e8fc5f38be3bab7ac6d37a35dcb56d31e62f7194da62b3a176593d1d8)

- JCE プロバイダー(SHA256 checksum a78ee59341da56af315d126ec7ed9f0dafe6e99649f659db2436be3863cb035b)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 18931bff869a0bd54846b3296d870fa19beba5652f979352be8fba6307e6d1aa)

Ubuntu 20.04 LTS

x86_64 アーキテクチャの Ubuntu 20.04 LTS 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum f1461c16b135ebcc17deec46aab88bd113ea122b8942fc188d4f05cd03e919a8)
- OpenSSL Dynamic Engine (SHA256 checksum 89211a7a7ed50eda2dc385c31ea76f1fbabd389ca691204873531d983c3eb0f7)
- JCE プロバイダー(SHA256 checksum f098c32d61a53b073459a75d88b68e377a9b16335874fae060cb10df0da00df0)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum ff91fb930717c917344af2ba344dc6e02bd5abc004dcb6147e9412b67e2aa7ab)

Windows Server 2022

x86_64 アーキテクチャの Windows Server 2022 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- JCE プロバイダー(SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- キーストレージプロバイダー (KSP) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Windows Server 2019

x86_64 アーキテクチャの Windows Server 2019 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- JCE プロバイダー(SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- キーストレージプロバイダー (KSP) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Windows Server 2016

x86_64 アーキテクチャの Windows Server 2016 用のバージョン 5.14.0 ソフトウェアを次のようにダウンロードします。

- PKCS #11 ライブラリ (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- JCE プロバイダー(SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
  - AWS CloudHSM (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb) 用の Javadocs
- CloudHSM CLI (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- キーストレージプロバイダー (KSP) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

クライアント SDK 5.14 では、CloudHSM CLI を使用したクォーラム制御によるキー使用およびキー管理オペレーションがサポートされました。クライアント SDK 5.14 には、Windows プラットフォームの AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP) のサポートも追加されています。さらに、クライアント SDK 5.14 では、Windows Server 2022 向けに、PKCS #11 ライブラリ、JCE プロバイダー、CloudHSM CLI、およびキーストレージプロバイダー (KSP) のインストールパッケージも追加されています。

プラットフォームのサポート

- PKCS #11 ライブラリ、JCE プロバイダー、CloudHSM CLI、およびキーストレージプロバイダー (KSP) について、Windows Server 2022 のサポートが追加されました。

CloudHSM CLI

- クォーラム制御によるキー使用およびキー管理オペレーションのサポートが追加されました。

キーストレージプロバイダー (KSP)

- Microsoft Windows オペレーティングシステム固有の暗号 API であるキーストレージプロバイダー (KSP) のサポートが追加されました。詳細については、AWS CloudHSM クライアント SDK 5 のキーストレージプロバイダー (KSP)を参照してください。

バージョン 5.13.0

クライアント SDK 5.13 で、hsm2m.medium クラスタータイプで相互 TLS 設定のサポートが追加されました。CloudHSM で相互 TLS を使用する方法については、「クライアントと の間の相互 TLS を設定する AWS CloudHSM (推奨)」を参照してください。クライアント SDK 5.13 では、Ubuntu 24.04 LTS のインストールパッケージも追加されています。

プラットフォームのサポート

- すべての SDK で x86_64 および ARM64 アーキテクチャでの Ubuntu 24.04 LTS のサポートが追加されました。

CloudHSM CLI

- 管理者ユーザーによる CloudHSM CLI でキーをレプリケートする コマンド実行のサポートが追加されました。クライアント SDK 5.12 で、Crypto User が使用するためのキーレプリケートコマンドが導入されました。
- 次のコマンドが追加されました。
  - CloudHSM CLI のクラスター mtls カテゴリ

バグ修正/機能向上

- クライアントが正常でない HSM 接続を検出するまでの時間を短縮するよう問題を修正しました。これにより、Lambda のウォームスタート時に発生する接続切断エラーを防止するのに役立ちます。

バージョン 5.12.0

クライアント SDK 5.12.0 で、複数のプラットフォームで ARM アーキテクチャに対応し、すべての SDK のパフォーマンスが改善されました。CloudHSM CLI と JCE プロバイダーに新機能が追加されました。

プラットフォームのサポート

- すべての SDK の ARM64 アーキテクチャで Amazon Linux 2023 のサポートが追加されました。
- すべての SDK の ARM64 アーキテクチャで Red Hat Enterprise Linux 9 (9.2 以降) のサポートが追加されました。
- すべての SDK の ARM64 アーキテクチャで Ubuntu 22.04 LTS のサポートが追加されました。

CloudHSM CLI

- 次のコマンドが追加されました。
  - CloudHSM CLI でキーをレプリケートする
- 複数のクラスターへの接続のサポートが追加されました。詳細については、「CloudHSM CLI を使用した複数のクラスターへの接続」を参照してください。

JCE プロバイダー

- `KeyStoreWithAttributes` を使用してキーを取得するために `KeyReferenceSpec` を追加しました。
- `KeyStoreWithAttributes` を使用して一度に複数のキーを取得するために `getKeys` を追加しました。

フォーマンスの向上

- すべての SDK の AES CBC NoPadding オペレーションのパフォーマンスが向上しました。

バージョン 5.11.0

クライアント SDK 5.11.0 では新しい機能が追加され、安定性が向上し、すべての SDK のバグ修正が含まれています。

プラットフォームのサポート

- すべての SDK について、Amazon Linux 2023 と RHEL 9 (9.2 以降) のサポートが追加されました。
- 最近サポート終了になったため、Ubuntu 18.04 LTS のサポートを削除しました。
- 最近サポート終了になったため、Amazon Linux のサポートを削除しました。

CloudHSM CLI

- 次のコマンドを追加しました。
  - CloudHSM CLI の暗号化署名カテゴリ
  - CloudHSM CLI の暗号化検証カテゴリ
  - CloudHSM CLI で PEM 形式キーをインポートする
  - CloudHSM CLI のキーアンラップコマンド
  - CloudHSM CLI のキーラップコマンド
- CloudHSM CLI で非対称キーをエクスポートする がパブリックキーのエクスポートをサポートするようになりました。

OpenSSL Dynamic Engine

- AWS CloudHSM OpenSSL Dynamic Engine は、OpenSSL ライブラリバージョン 3.x にインストールされているプラットフォームでサポートされるようになりました。これには、Amazon Linux 2023、RHEL 9 (9.2 以降)、Ubuntu 22.04 が含まれます。

JCE

- JDK 17 と JDK 21 のサポートが追加されました
- HMAC オペレーションに使用される AES キーのサポートが追加されました。
- 新しいキー属性 `ID` を追加しました。

- キーを使い果たすことに対する新しい `DataExceptionCause` バリアントを導入しました: `DataExceptionCause.KEY_EXHAUSTED`。

バグ修正/機能向上

- `label` 属性の最大長を 126 文字から 127 文字に増やしました。
- RsaOaep メカニズムを使用した EC キーのラップ解除ができないバグを修正しました。
- JCE プロバイダーの getKey オペレーションの既知の問題を解決しました。詳細については、問題: getKey 操作によるクライアント SDK 5 メモリリーク を参照してください。
- FIPS 140-2 に従って、暗号化ブロックの最大制限に達した Triple DES キーのすべての SDK の口グ記録が改善されました。
- OpenSSL Dynamic Engine の既知の問題を追加しました。詳細については、「の OpenSSL Dynamic Engine の既知の問題 AWS CloudHSM」を参照してください。

バージョン 5.10.0

クライアント SDK 5.10.0 では安定性が向上し、すべての SDK のバグ修正が含まれています。

CloudHSM CLI

- 顧客が CloudHSM CLI を使用してキーを管理できるようにする次のような新しいコマンドを追加しました。
  - 対称キーと非対称キーペアを作成
  - 共有と共有解除キー
  - キー属性を使用してキーを一覧表示およびフィルタリングします
  - キー属性の設定
  - キーリファレンスファイルの生成
  - キーの削除
- エラーログ記録を改良しました。
- インタラクティブモードでの複数行の Unicode コマンドのサポートが追加されました。

バグ修正/機能向上

- すべての SDK のセッションキーのインポート、ラップ解除、派生、作成のパフォーマンスが向上しました。

- 終了時に一時ファイルが削除されないという JCE プロバイダのバグを修正しました。
- クラスター内の HSM が置き換えられた後、特定の条件下で接続エラーが発生するバグを修正しました。
- 大きなマイナーバージョン番号を処理し、パッチ番号を含むように JCE `getVersion` 出力形式を変更しました。

プラットフォームのサポート

- JCE、PKCS #11、および CloudHSM CLI を搭載した Ubuntu 22.04 のサポートを追加しました (OpenSSL ダイナミックエンジンのサポートはまだありません)。

バージョン 5.9.0

クライアント SDK 5.9.0 では安定性が向上し、すべての SDK のバグ修正が含まれています。すべての SDK が最適化され、HSM が使用できないと判断されるとすぐにアプリケーションにオペレーションの失敗を通知できるようになりました。このリリースには JCE のパフォーマンス強化が含まれています。

JCE プロバイダー

- パフォーマンスの向上
- セッションプールの枯渇に関する既知の問題が修正されました

AWS CloudHSM 非推奨の Client SDK リリース

5.8.0 以前のバージョンは非推奨になりました。本番ワークロードでは、非推奨のリリースを使用することはお勧めしません。非推奨のリリースに下位互換性のあるアップデートを提供したり、ダウンロード用に非推奨のリリースをホストしたりすることはありません。非推奨のリリースの使用中に本番環境への影響が発生した場合は、アップグレードしてソフトウェアフィックスを入手する必要があります。

非推奨のクライアント SDK 5 リリース

このセクションでは、非推奨のクライアント SDK 5 リリースを一覧表示します。

バージョン 5.8.0

バージョン5.8.0では、CloudHSM CLI のクォーラム認証、JSSEによるSSL/TLSオフロード、PKCS #11 のマルチスロットサポート、JCEのマルチクラスター/マルチユーザーサポート、JCEによるキー抽出、KeyFactory for JCE、非端末リターンコードの新しい再試行構成が導入され、すべての SDK の安定性の向上とバグ修正が含まれています。

PKCS #11 ライブラリ

- マルチスロット構成のサポートが追加されました。

JCE プロバイダー

- 設定ベースのキー抽出が追加されました。
- マルチクラスター構成とマルチユーザー構成のサポートが追加されました。
- JSSE による SSL と TLS のオフロードのサポートが追加されました。
- AES/CBC/NoPadding のアンラップサポートが追加されました。
- シークレットキーファクトリとキーファクトリという新しいタイプのキーファクトリが追加されました。

CloudHSM CLI

- クォーラム認証のサポートを追加する

バージョン 5.7.0

バージョン 5.7.0 では CloudHSM CLI が導入され、新しい暗号ベースのメッセージ認証コード (CMAC) アルゴリズムが含まれています。このリリースでは、Amazon Linux 2 に ARM アーキテクチャが追加されました。JCE プロバイダー Javadoc が AWS CloudHSMで利用できるようになりました。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。
- Amazon Linux 2 の ARM アーキテクチャでサポートされるようになりました。
- アルゴリズム
  - CKM_AES_CMAC (署名して確認)

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。
- Amazon Linux 2 の ARM アーキテクチャでサポートされるようになりました。

JCE プロバイダー

- 安定性の向上およびバグ修正。
- アルゴリズム
  - AESCMAC

バージョン 5.6.0

バージョン 5.6.0 には、PKCS #11 ライブラリと JCE プロバイダーの新しいメカニズムサポートが含まれています。さらに、バージョン 5.6 は Ubuntu 20.04 をサポートしています。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。
- メカニズム
  - CKM_RSA_X_509、暗号化、復号化、署名、検証の各モード用

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- 安定性の向上およびバグ修正。
- 暗号
  - RSA/ECB/NOPADDING (暗号化モードと復号モード用)

サポートされるキー

- 曲線 secp224r1 と secp521r1 の EC

プラットフォームのサポート

- Ubuntu 20.04 に追加されたサポート。

バージョン 5.5.0

バージョン5.5.0では、OpenJDK 11、キーツールとJarsignerの統合、およびJCEプロバイダへの追加メカニズムのサポートが追加されています。KeyGenerator クラスがキーサイズパラメーターをビット数ではなくバイト数として誤って解釈するという 既知の問題 が解決されました。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- Keytool ユーティリティと Jarsigner ユーティリティのサポート
- すべてのプラットフォームでの OpenJDK 11 のサポート
- 暗号
  - AES/CBC/NOPadding (暗号化モードと復号化モード)
  - AES/ECB/PKCs5Padding (暗号化モードと復号化モード)
  - AES/CTR/NO/PADDING (暗号化モードと復号化モード)
  - AES/GCM/noPadding (ラップ/アンラップモード)
  - desede/ECB/PKCS5Padding (暗号化モードと復号化モード)
  - desede/CBC/NOPADDING (暗号化モードと復号化モード)
  - AESWrap/ECB/noPadding (ラップ/アンラップモード)
  - AESWrap/ECB/PKCS5Padding (ラップ/アンラップモード)
  - AESWrap/ECB/ZeroPadding (ラップ/アンラップモード)
  - RSA/ECB/PKCS1-Padding (ラップ/アンラップモード)
  - RSA/ECB/OAEPPadding (ラップ/アンラップモード)
  - RSA/ECB/OAEP (SHA-1 と MGF1 パディング、ラップ/アンラップモード)

- RSA/ECB/OAEP (SHA-224 と MGF1 パディング、ラップ/アンラップモード)
- RSA/ECB/OAEP (SHA-256 と MGF1 パディング、ラップ/アンラップモード)
- RSA/ECB/OAEP (SHA-384 と MGF1 パディング、ラップ/アンラップモード)
- RSA/ECB/OAEP (SHA-512 と MGF1 パディング、ラップ/アンラップモード)
- RSAAESWrap/ECB/OAEPPadding (ラップ/アンラップモード)
- RSaaES Wrap/ECB/OAEP (SHA-1 と MGF1 パディング、ラップ/アンラップモード)
- RSaaES Wrap/ECB/OAEP (SHA-224 と MGF1 パディング、ラップ/アンラップモード)
- RSaaES Wrap/ECB/OAEP (SHA-256 と MGF1 パディング、ラップ/アンラップモード)
- RSaaES Wrap/ECB/OAEP (SHA-384 と MGF1 パディング、ラップ/アンラップモード)
- RSaaES Wrap/ECB/OAEP (SHA-512 と MGF1 パディング、ラップ/アンラップモード)

- キーファクトリとシークレットキーファクトリ
  - RSA – 2,048 ~ 4,096 ビットの RSA キー (256 ビットの増分)
  - AES – 128、192、256 ビットの AES キー
  - NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384)、および secp256k1 を対象とした EC キーペア
  - DeSede (3DES)
  - ジェネリック・シークレット
  - HMAC – SHA1、SHA224、SHA256、SHA384、SHA512 ハッシュをサポート
- 署名/検証
  - RSASSA-PSS
  - SHA1withRSA/PSS
  - SHA224withRSA/PSS
  - SHA256withRSA/PSS
  - SHA384withRSA/PSS
  - SHA512withRSA/PSS
  - SHA1withRSAandMGF1
  - SHA224withRSAandMGF1
  - SHA256withRSAandMGF1
  - SHA384withRSAandMGF1
  - SHA512withRSAandMGF1

バージョン 5.4.2

バージョン 5.4.2 では、すべての SDK の安定性が向上し、バグが修正されています。これは CentOS 8 プラットフォームの最後のリリースでもあります。詳細については、「CentOS のウェブサイト」を参照してください。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- 安定性の向上およびバグ修正。

バージョン 5.4.1

バージョン 5.4.1 では、PKCS #11 ライブラリの既知の問題が解決されました。これは CentOS 8 プラットフォームの最後のリリースでもあります。詳細については、「CentOS のウェブサイト」を参照してください。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- 安定性の向上およびバグ修正。

バージョン 5.4.0

バージョン 5.4.0 では、すべてのプラットフォームの JCE プロバイダーの初期サポートが追加されています。JCE プロバイダーは OpenJDK 8 と互換性があります。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- キータイプ
  - RSA – 2,048 ~ 4,096 ビットの RSA キー (256 ビットの増分)。
  - AES – 128、192、256 ビットの AES キー。
  - NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384)、および secp256k1 を対象とした ECC キーペア。
  - DeSede (3DES)
  - HMAC – SHA1、SHA224、SHA256、SHA384、SHA512 ハッシュをサポート。
- 暗号 (暗号化と復号のみ)
  - AES/GCM/NoPadding
  - AES/ECB/NoPadding
  - AES/CBC/PKCS5Padding
  - デシード/ECB/パディングなし
  - スウェーデン/CBC/PKCS5 パディング
  - AES/CTR/NoPadding
  - RSA/ECB/PKCS1Padding
  - RSA/ECB/OAEPPadding
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding

  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
- ダイジェスト
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- 署名/検証
  - NONEwithRSA
  - SHA1withRSA
  - SHA224withRSA
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1 (ECDSA 搭載)
  - SHA224 (ECDSA 搭載)
  - SHA256 (ECDSA 搭載)
  - SHA384 (ECDSA 搭載)
  - SHA512 (ECDSA 搭載)
- Java KeyStore との統合

バージョン 5.3.0

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 曲線 P-256、P-384、secp256k1 による ECDSA 署名/検証のサポートを追加します。

- Amazon Linux、Amazon Linux 2、CentOS 7.8 以降、RHEL 7 (7.8 以降) のプラットフォームのサポートが追加されました。
- OpenSSL バージョン 1.0.2 のサポートが追加されました。
- 安定性の向上およびバグ修正。

JCE プロバイダー

- キータイプ
  - RSA – 2,048 ~ 4,096 ビットの RSA キー (256 ビットの増分)。
  - AES – 128、192、256 ビットの AES キー。
  - NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384)、および secp256k1 を対象とした EC キーペア。
  - DeSede (3DES)
  - HMAC – SHA1、SHA224、SHA256、SHA384、SHA512 ハッシュをサポート。
- 暗号 (暗号化と復号のみ)
  - AES/GCM/NoPadding
  - AES/ECB/NoPadding
  - AES/CBC/PKCS5Padding
  - デシード/ECB/パディングなし
  - スウェーデン/CBC/PKCS5 パディング
  - AES/CTR/NoPadding
  - RSA/ECB/PKCS1Padding
  - RSA/ECB/OAEPPadding
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
- ダイジェスト
  - SHA-1
  - SHA-224

  - SHA-256
  - SHA-384
  - SHA-512
- 署名/検証
  - NONEwithRSA
  - SHA1withRSA
  - SHA224withRSA
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1 (ECDSA 搭載)
  - SHA224 (ECDSA 搭載)
  - SHA256 (ECDSA 搭載)
  - SHA384 (ECDSA 搭載)
  - SHA512 (ECDSA 搭載)
- Java KeyStore との統合

バージョン 5.2.1

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

バージョン 5.2.0

バージョン 5.2.0 では、PKCS #11 ライブラリの追加のキーのタイプとメカニズムのサポートが追加されました。

PKCS #11 ライブラリ

キーのタイプ

- ECDSA– P-224、P-256、P-384、P-521、および楕円曲線暗号 secp256k1
- Triple DES (3DES)

メカニズム

- CKM_EC_KEY_PAIR_GEN
- CKM_DES3_KEY_GEN
- CKM_DES3_CBC
- CKM_DES3_CBC_PAD
- CKM_DES3_ECB
- CKM_ECDSA
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512
- 暗号化および復号用 CKM_RSA_PKCS

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

バージョン 5.1.0

バージョン 5.1.0 では、PKCS #11 ライブラリの追加のメカニズムのサポートが追加されました。

PKCS #11 ライブラリ

メカニズム

- ラップおよびラップ解除用 CKM_RSA_PKCS
- CKM_RSA_PKCS_PSS

- CKM_SHA1_RSA_PKCS_PSS
- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_SP800_108_COUNTER_KDF
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_SHA_1_HMAC
- CKM_SHA224_HMAC
- CKM_SHA256_HMAC
- CKM_SHA384_HMAC
- CKM_SHA512_HMAC
- CKM_RSA_PKCS_OAEP ラップおよびラップ解除のみ
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

API オペレーション

- C_CreateObject
- C_DeriveKey
- C_WrapKey
- C_UnWrapKey

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

バージョン 5.0.1

バージョン 5.0.1 では、OpenSSL Dynamic Engine の初期サポートが追加されました。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- OpenSSL Dynamic Engine の初期リリース
- このリリースでは、キーのタイプと OpenSSL API の入門サポートを提供しています。
  - 2,048、3,072、および 4,096 ビットキーの RSA キーの生成
  - OpenSSL API:
    - SHA1/224/256/384/512 と RSA PSS で RSA PKCS を使った RSA サイン
    - RSA キーの生成

  詳細については、「OpenSSL Dynamic Engine」を参照してください。
- 対応プラットフォーム: CentOS 8.3+、Red Hat Enterprise Linux (RHEL) 8.3+、Ubuntu 18.04 LTS
  - 必要事項: OpenSSL 1.1.1

  詳細については、「Supported Platforms (サポートされているプラットフォーム)」を参照してください。
- NGINX 1.19 を含む、CentOS 8.3+、Red Hat Enterprise Linux (RHEL) 8.3、Ubuntu 18.04 LTS での SSL/TLS のオフロードのサポート (一部の暗号スイート向け)。

  詳細については、「Tomcat を使用した Linux での SSL/TLS オフロード」または「NGINX または Apache を使用した Linux での SSL/TLS オフロード」を参照してください。

バージョン 5.0.0

バージョン 5.0.0 が最初のリリースです。

PKCS #11 ライブラリ

- これは最初のリリースです。

クライアント SDK バージョン 5.0.0 での PKCS #11 ライブラリ入門サポート

このセクションでは、クライアント SDK バージョン 5.0.0 のキーのタイプ、メカニズム、API オペレーション、および属性のサポートについて詳しく説明します。

キーのタイプ

- AES - 128、192、256 ビットの AES キー
- RSA - 2,048 ~ 4,096 ビットの RSA キー (256 ビットの増分)

メカニズム

- CKM_AES_GCM
- CKM_AES_KEY_GEN
- CKM_CLOUDHSM_AES_GCM
- CKM_RSA_PKCS
- CKM_RSA_X9_31_KEY_PAIR_GEN
- CKM_SHA1
- CKM_SHA1_RSA_PKCS
- CKM_SHA224
- CKM_SHA224_RSA_PKCS
- CKM_SHA256
- CKM_SHA256_RSA_PKCS
- CKM_SHA384
- CKM_SHA384_RSA_PKCS
- CKM_SHA512
- CKM_SHA512_RSA_PKCS

API オペレーション:

- C_CloseAllSessions
- C_CloseSession
- C_Decrypt

- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DestroyObject
- C_Digest
- C_DigestFinal
- C_DigestInit
- C_DigestUpdate
- C_Encrypt
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate
- C_Finalize
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize

- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal
- C_SignInit
- C_SignUpdate
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyUpdate

属性:

- GenerateKeyPair
  - すべての RSA キー属性
- GenerateKey
  - すべての AES キー属性
- GetAttributeValue
  - すべての RSA キー属性
  - すべての AES キー属性

サンプル

- キーの生成 (AES、RSA、EC)
- キー属性のリスト化
- AES GCM を使用したデータの暗号化および復号
- RSAを使用したデータの署名と検証

非推奨のクライアント SDK 3 リリース

このセクションでは、非推奨のクライアント SDK 3 リリースを一覧表示します。

バージョン 3.4.4

バージョン 3.4.4 では、JCE プロバイダーに更新が追加されます。

AWS CloudHSM クライアントソフトウェア

- 整合性のために更新されたバージョン。

PKCS #11 ライブラリ

- 整合性のために更新されたバージョン。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。

JCE プロバイダー

- log4j をバージョン 2.17.1 に更新します。

Windows (CNG および KSG プロバイダー)

- 整合性のために更新されたバージョン。

バージョン 3.4.3

バージョン 3.4.3 では、JCE プロバイダーに更新が追加されます。

AWS CloudHSM クライアントソフトウェア

- 整合性のために更新されたバージョン。

PKCS #11 ライブラリ

- 整合性のために更新されたバージョン。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。

JCE プロバイダー

- log4j をバージョン 2.17.0 に更新します。

Windows (CNG および KSG プロバイダー)

- 整合性のために更新されたバージョン。

バージョン 3.4.2

バージョン 3.4.2 では、JCE プロバイダーに更新が追加されます。

AWS CloudHSM クライアントソフトウェア

- 整合性のために更新されたバージョン。

PKCS #11 ライブラリ

- 整合性のために更新されたバージョン。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。

JCE プロバイダー

- log4j をバージョン 2.16.0 に更新します。

Windows (CNG および KSG プロバイダー)

- 整合性のために更新されたバージョン。

バージョン 3.4.1

バージョン 3.4.1 では、JCE プロバイダーに更新が追加されます。

AWS CloudHSM クライアントソフトウェア

- 整合性のために更新されたバージョン。

PKCS #11 ライブラリ

- 整合性のために更新されたバージョン。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。

JCE プロバイダー

- log4j をバージョン 2.15.0 に更新します。

Windows (CNG および KSG プロバイダー)

- 整合性のために更新されたバージョン。

バージョン 3.4.0

バージョン 3.4.0 では、すべてのコンポーネントに更新が追加されます。

AWS CloudHSM クライアントソフトウェア

- 安定性の向上およびバグ修正。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- 安定性の向上およびバグ修正。

Windows (CNG および KSG プロバイダー)

- 安定性の向上およびバグ修正。

バージョン 3.3.2

バージョン 3.3.2 では問題解決に client_info スクリプトを使用します。

AWS CloudHSM クライアントソフトウェア

- 整合性のために更新されたバージョン。

PKCS #11 ライブラリ

- 整合性のために更新されたバージョン。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。

JCE プロバイダー

- 整合性のために更新されたバージョン。

Windows (CNG および KSG プロバイダー)

- 整合性のために更新されたバージョン。

バージョン 3.3.1

バージョン 3.3.1 では、すべてのコンポーネントに更新が追加されます。

AWS CloudHSM クライアントソフトウェア

- 安定性の向上およびバグ修正。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- 安定性の向上およびバグ修正。

Windows (CNG および KSG プロバイダー)

- 安定性の向上およびバグ修正。

バージョン 3.3.0

バージョン 3.3.0 では、2 要素認証 (2FA) の追加などの改良が行われました。

AWS CloudHSM クライアントソフトウェア

- Crypto Officer (CO) の 2FA 認証を追加しました。詳細については、「Managing Two-Factor Authentication for Crypto Officers (Crypto Officer 用の 2 要素認証の管理)」を参照してください。
- RedHat Enterprise Linux 6 および CentOS 6 用のプラットフォームのサポートを削除しました。詳細については、「Linux サポート」を参照してください。
- クライアント SDK 5 またはクライアント SDK 3 で使用する独立型の CMU を追加しました。これは、バージョン 3.3.0 のクライアントデーモンに含まれている CMU のバージョンと同じで、クライアントデーモンをダウンロードせずに CMU をダウンロードできるようになりました。

PKCS #11 ライブラリ

- 安定性の向上およびバグ修正。
- RedHat Enterprise Linux 6 および CentOS 6 用のプラットフォームのサポートを削除しました。詳細については、「Linux サポート」を参照してください。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。
- RedHat Enterprise Linux 6 および CentOS 6 用のプラットフォームのサポートを削除しました。詳細については、「Linux サポート」を参照してください。

JCE プロバイダー

- 安定性の向上およびバグ修正。
- RedHat Enterprise Linux 6 および CentOS 6 用のプラットフォームのサポートを削除しました。詳細については、「Linux サポート」を参照してください。

Windows (CNG および KSG プロバイダー)

- 整合性のために更新されたバージョン。

バージョン 3.2.1

バージョン 3.2.1 では、PKCS #11 ライブラリの AWS CloudHSM 実装と PKCS #11 標準、新しいプラットフォーム、その他の改善点の間のコンプライアンス分析が追加されています。

AWS CloudHSM クライアントソフトウェア

- CentOS 8、RHEL 8、Ubuntu 18.04 LTS 用のプラットフォームのサポートを追加します。詳細については、「???」を参照してください。

PKCS #11 ライブラリ

- クライアント SDK 3.2.1 用 PKCS #11 ライブラリコンプライアンスレポート
- CentOS 8、RHEL 8、Ubuntu 18.04 LTS 用のプラットフォームのサポートを追加します。詳細については、「???」を参照してください。

OpenSSL Dynamic Engine

- CentOS 8、RHEL 8、Ubuntu 18.04 LTS 用のサポートがありません。詳細については、「???」を参照してください。

JCE プロバイダー

- CentOS 8、RHEL 8、Ubuntu 18.04 LTS 用のプラットフォームのサポートを追加します。詳細については、「???」を参照してください。

Windows (CNG および KSG プロバイダー)

- 安定性の向上およびバグ修正。

バージョン 3.2.0

バージョン 3.2.0 では、パスワードのマスキングのサポートやその他の改善点が追加されました。

AWS CloudHSM クライアントソフトウェア

- コマンドラインツールを使用するときにパスワードを非表示にするサポートが追加されました。詳細については、「loginHSM および logoutHSM (CloudHSM _mgmt_util)」と「loginHSM および logoutHSM (key_mgmt_util)」を参照してください。

PKCS #11 ライブラリ

- 以前サポートされていなかった一部の PKCS #11 メカニズムについて、ソフトウェアでラージデータをハッシュするためのサポートが追加されました。詳細については、「Supported Mechanisms (サポートされているメカニズム)」を参照してください。

OpenSSL Dynamic Engine

- 安定性の向上およびバグ修正。

JCE プロバイダー

- 整合性のために更新されたバージョン。

Windows (CNG および KSG プロバイダー)

- 安定性の向上およびバグ修正。

バージョン 3.1.2

バージョン 3.1.2 では、JCE プロバイダーに更新が追加されます。

AWS CloudHSM クライアントソフトウェア

- 整合性のために更新されたバージョン。

PKCS #11 ライブラリ

- 整合性のために更新されたバージョン。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。

JCE プロバイダー

- log4j をバージョン 2.13.3 に更新します。

Windows (CNG および KSG プロバイダー)

- 整合性のために更新されたバージョン。

バージョン 3.1.1

AWS CloudHSM クライアントソフトウェア

- 整合性のために更新されたバージョン。

PKCS #11 ライブラリ

- 整合性のために更新されたバージョン。

OpenSSL Dynamic Engine

- 整合性のために更新されたバージョン。

JCE プロバイダー

- パフォーマンス向上とバグ修正が行われています。

Windows (CNG、KSP)

- 整合性のために更新されたバージョン。

バージョン 3.1.0

バージョン 3.1.0 では、標準に準拠した AES キーラップが追加されました。

AWS CloudHSM クライアントソフトウェア

- アップグレードの新しい要件: クライアントのバージョンは、使用しているソフトウェアライブラリのバージョンと一致する必要があります。アップグレードするには、クライアントとすべてのライブラリを同時にアップグレードするバッチコマンドを使用する必要があります。詳細については、「クライアント SDK 3 のアップグレード」を参照してください。
- Key_mgmt_util (KMU) には次の更新が含まれています。
  - 2 つの新しい AES キーラップ方法が追加されました。標準に準拠したゼロパディングを使用する AES キーラップと、パディングなしの AES キーラップです。詳細については、「wrapKey」および「unwrapKey」を参照してください。
  - AES_KEY_WRAP_PAD_PKCS5 を使用してキーをラップするときにカスタム IV を指定する機能が無効になりました。詳細については、「AES キーラップ」を参照してください。

PKCS #11 ライブラリ

- 2 つの新しい AES キーラップ方法が追加されました。標準に準拠したゼロパディングを使用する AES キーラップと、パディングなしの AES キーラップです。詳細については、「AES キーラップ」を参照してください。
- RSA-PSS 署名のソルトの長さを設定できます。この機能の使用方法については、GitHub の「設定可能な RSA-PSS 署名のソルトの長さ」を参照してください。

OpenSSL Dynamic Engine

- 重要な変更: SHA1 を使用する TLS 1.0 および 1.2 暗号スイートは、OpenSSL Engine 3.1.0 では利用できません。この問題は間もなく解決されます。
- RHEL 6 または CentOS 6 に OpenSSL Dynamic Engine ライブラリをインストールする場合は、これらのオペレーティングシステムにインストールされているデフォルトの OpenSSL バージョンに関する既知の問題を参照してください。
- 安定性の向上およびバグ修正。

JCE プロバイダー

- 重要な変更: Java Cryptography Extension (JCE) 準拠の問題に対処するため、AES ラップとラップ解除で AES アルゴリズムの代わりに AES Wrap アルゴリズムが適切に使用されるようになりました。つまり、AES/ECB および AES/CBC メカニズムでは、`Cipher.WRAP_MODE` および `Cipher.UNWRAP_MODE` は成功しなくなりました。

  クライアントバージョン 3.1.0 にアップグレードするには、コードを更新する必要があります。既存のラップされたキーがある場合は、ラップ解除に使用するメカニズムと IV のデフォルトの変更方法に特に注意する必要があります。クライアントバージョン 3.0.0 以前でキーをラップした場合、3.1.1 では、AESWrap/ECB/PKCS5Padding を使用して既存のキーをラップ解除する必要があります。詳細については、「AES キーラップ」を参照してください。
- JCE プロバイダーから同じラベルを持つ複数のキーを一覧表示できます。使用可能なすべてのキーを反復処理する方法については、GitHub の「すべてのキーを検索する」を参照してください。
- パブリックキーとプライベートキーに異なるラベルを指定するなど、キーの作成時に属性に対してより制限的な値を設定できます。詳細については、「サポートされている Java 属性」を参照してください。

Windows (CNG、KSP)

- 安定性の向上およびバグ修正。

クライアント SDK AWS CloudHSM end-of-lifeリリース

次の AWS CloudHSM クライアントバージョンはサポートが終了しました。これらの AWS CloudHSM クライアントバージョンは、サービスとの互換性がなく、今後更新プログラムを受け取ることはありません。アプリケーションのセキュリティを維持するために、サポートが終了したリリースからの接続を拒否 AWS CloudHSM する可能性があります。

- SDK バージョン 3.4.4 以前のバージョンはサポートが終了しました。
- SDK バージョン 5.8.0 以前のバージョンはサポートが終了しました。

ドキュメント履歴

このトピックでは、AWS CloudHSM ユーザーガイドの重要な更新について説明します。

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 4 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、右上隅にあるリンクを使用して、RSS フィードをサブスクライブします。

新しいリリースの詳細については、「AWS CloudHSM クライアント SDK のダウンロード」を参照してください

変更	説明	日付
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.18.0 をリリースしました。	2026 年 8 月 18 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.17.2 をリリースしました。	2026 年 5 月 22 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.17.1 をリリースしました。	2026 年 3 月 19 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.17.0 をリリースしました。	2026 年 1 月 6 日

OpenSSL プロバイダーのダウンロードリンクを追加	AWS CloudHSM クライアントバージョン 5.17 への OpenSSL プロバイダーのダウンロードリンクを追加しました。	2026 年 1 月 6 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.16.2 をリリースしました。	2025 年 9 月 16 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.16.1 をリリースしました。	2025 年 6 月 25 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.16.0 をリリースしました。	2025 年 5 月 1 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.15.0 をリリースしました。	2025 年 2 月 3 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.14.0 をリリースしました。	2024 年 11 月 26 日
新しい HSM タイプとクラスターモード	FIPS モードクラスターで (hsm2m.medium) を作成するサポートが追加されました。	2024 年 8 月 20 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.13.0 をリリースしました。	2024 年 8 月 13 日
新しい HSM タイプとクラスターモード	新しい HSM タイプ (hsm2m.medium) と新しいクラスターモード (非 FIPS) を起動しました。	2024 年 6 月 10 日

新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.12.0 をリリースしました。	2024 年 3 月 20 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.11.0 をリリースしました。	2024 年 1 月 17 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.10.0 をリリースしました。	2023 年 7 月 28 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.9.0 をリリースしました。	2023 年 5 月 23 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.8.0 をリリースしました。	2023 年 3 月 16 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.7.0 をリリースしました。	2022 年 11 月 16 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.6.0 をリリースしました。	2022 年 9 月 1 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.5.0 をリリースしました。	2022 年 5 月 13 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.4.2 をリリースしました。	2022 年 3 月 18 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.4.1 をリリースしました。	2022 年 2 月 10 日

新しいリリースの追加	Windows プラットフォーム用の AWS CloudHSM JCE プロバイダーバージョン 5.4.0 をリリースしました。	2022 年 2 月 1 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.4.0 をリリースしました。これにより、すべての Linux プラットフォームの JCE プロバイダーの初期サポートが追加されました。	2022 年 1 月 28 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.3.0 をリリースしました。	2022 年 1 月 3 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.4.4 をリリースしました。	2022 年 1 月 3 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.4.3 をリリースしました。	2021 年 12 月 20 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.4.2 をリリースしました。	2021 年 12 月 15日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.4.1 をリリースしました。	2021 年 12 月 10 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.2.1 をリリースしました。	2021 年 10 月 4 日

新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.4.0 をリリースしました。	2021 年 8 月 25 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.2.0 をリリースしました。	2021 年 8 月 3 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.3.2 をリリースしました。	2021 年 7 月 2 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.1.0 をリリースしました。	2021 年 6 月 1 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.3.1 をリリースしました。	2021 年 4 月 26 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.0.1 をリリースしました。	2021 年 4 月 8 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 5.0.0 をリリースしました。	2021 年 3 月 12 日
新しいコンテンツの追加	インターネット、NAT デバイス、VPN 接続、または 接続を介したアクセスを必要と AWS CloudHSM せずに、VPC と の間にプライベート Direct Connect 接続を作成できる AWS 機能であるインターフェイス VPC エンドポイントを追加しました。	2021 年 2 月 10 日

新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.3.0 をリリースしました。	2021 年 2 月 3 日
新しいコンテンツの追加	古いバックアップを自動的に削除する機能である、マネージドバックアップ保持機能が追加されました。	2020 年 11 月 18 日
新しいコンテンツの追加	PKCS #11 標準を使用して PKCS #11 ライブラリの AWS CloudHSM クライアント SDK 3.2.1 実装を分析するコンプライアンスレポートを追加しました。	2020 年 10 月 29 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.2.1 をリリースしました。	2020 年 10 月 8 日
新しいコンテンツの追加	AWS CloudHSMのキーの同期設定について説明したドキュメントを追加しました。	2020 年 9 月 1 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.2.0 をリリースしました。	2020 年 8 月 31 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.1.2 をリリースしました。	2020 年 7 月 30 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.1.1 をリリースしました。	2020 年 6 月 3 日

新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.1.0 をリリースしました。	2020 年 5 月 21 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 3.0.1 をリリースしました。	2020 年 4 月 20 日
新しいリリースの追加	Windows Server プラットフォーム用の AWS CloudHSM クライアントバージョン 3.0.0 をリリースしました。	2019 年 10 月 30 日
新しいリリースの追加	Windows を除くすべてのプラットフォーム用の AWS CloudHSM クライアントバージョン 3.0.0 をリリースしました。	2019 年 10 月 22 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 2.0.4 をリリースしました。	2019 年 8 月 26 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 2.0.3 をリリースしました。	2019 年 5 月 13 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 2.0.1 をリリースしました。	2019 年 3 月 21 日
新しいリリースの追加	AWS CloudHSM クライアントバージョン 2.0.0 をリリースしました。	2019 年 2 月 6 日

リージョンサポートの追加	欧州 (ストックホルム) および AWS GovCloud (米国東部) リージョン AWS CloudHSM のサポートが追加されました。	2018 年 12 月 19 日
新しいリリースの追加	Windows 用の AWS CloudHSM クライアントバージョン 1.1.2 をリリースしました。	2018 年 11 月 20 日
更新された既知の問題	トラブルシューティングガイドに新しいコンテンツが追加されました。	2018 年 11 月 8 日
新しいリリースの追加	Linux プラットフォーム用の AWS CloudHSM クライアントバージョン 1.1.2 をリリースしました。	2018 年 11 月 8 日
リージョンサポートの追加	欧州 (パリ) およびアジアパシフィック (ソウル) リージョン AWS CloudHSM のサポートを追加しました。	2018 年 10 月 24 日
新しいコンテンツの追加	AWS CloudHSM バックアップを削除および復元する機能を追加しました。	2018 年 9 月 10 日
新しいコンテンツの追加	Amazon CloudWatch Logs への自動監査ログ配信を追加しました。	2018 年 8 月 13 日
新しいコンテンツの追加	リージョン間で AWS CloudHSM クラスターバックアップをコピーする機能を追加しました。	2018 年 7 月 30 日

リージョンサポートの追加	欧州 (ロンドン) リージョン AWS CloudHSM のサポートを追加しました。	2018 年 6 月 13 日
新しいコンテンツの追加	Amazon Linux 2、Red Hat Enterprise Linux (RHEL) 6、Red Hat Enterprise Linux (RHEL) 7、CentOS 6、CentOS 7、Ubuntu 16.04 LTS の AWS CloudHSM クライアントとライブラリのサポートを追加しました。	2018 年 5 月 10 日
新しいリリースの追加	Windows AWS CloudHSM クライアントを追加しました。	2018 年 30 月 4 日

以前の更新

次の表は、2018 AWS CloudHSM 年以前の の重要な変更点を示しています。

変更	説明	日付
新しいコンテンツ	Crypto User (CO) 用のクォーラム認証 (M of N アクセスコントロール) を追加しました。詳細については、「CloudHSM 管理ユーティリティ (CMU) を使用したクォーラム認証の管理 (M of N アクセスコントロール)」を参照してください。	2017 年 11 月 9 日
更新	key_mgmt_util コマンドラインツールの使用方法についてのドキュメントを追加しました。詳細については、「AWS CloudHSM キー管理ユーティ	2017 年 11 月 9 日

変更	説明	日付
	リティコマンドのリファレンス」を参照してください。	
新しいコンテンツ	Oracle Transparent Data Encryption を追加しました。詳細については、「Oracle Database 暗号化」を参照してください。	2017 年 10 月 25 日
新しいコンテンツ	SSL オフロードを追加しました。詳細については、「SSL/TLS のオフロード」を参照してください。	2017 年 10 月 12 日
新規ガイド	このリリースでは、 AWS CloudHSM	2017 年 8 月 14 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。